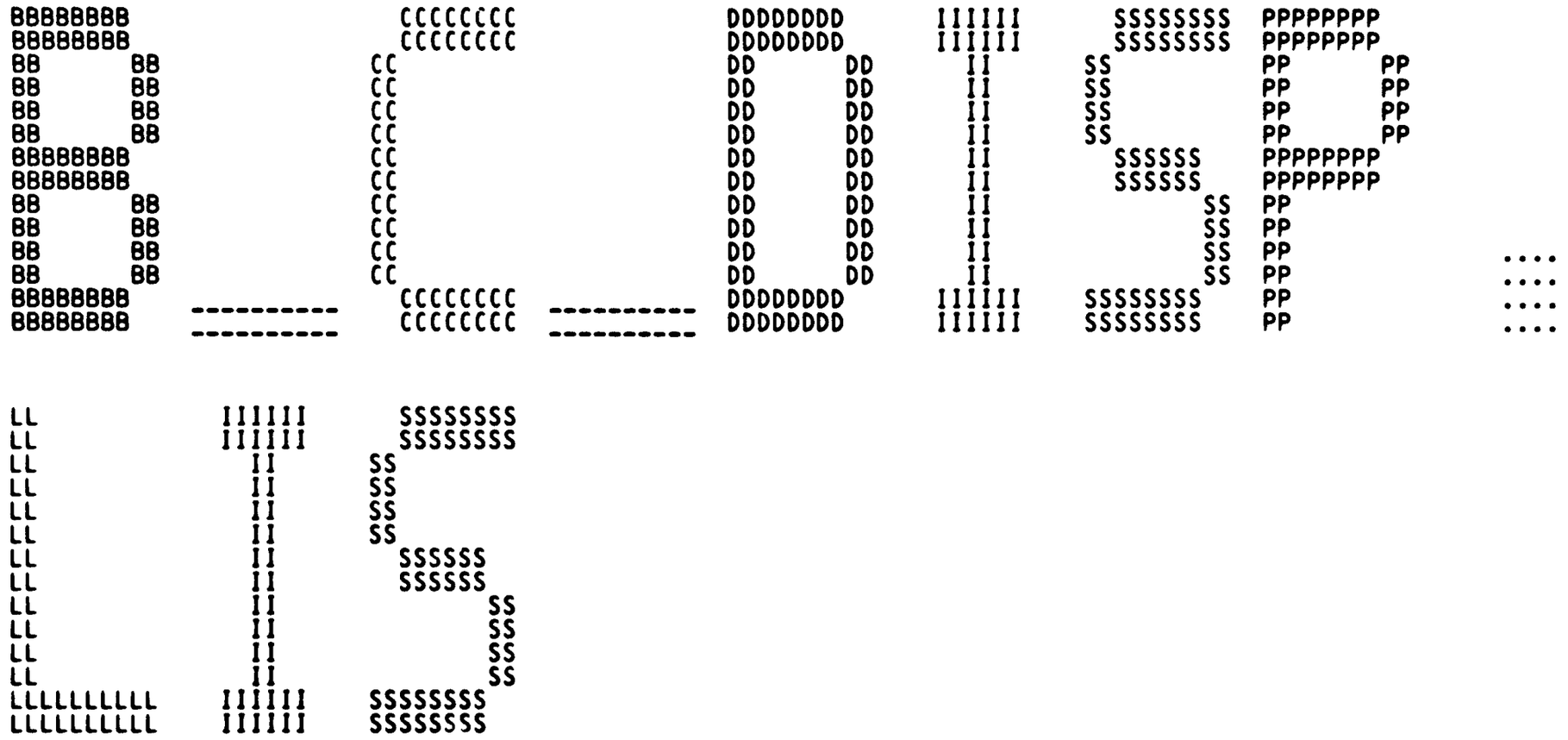
```
EEEEEEEEEEEEEEE  RRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR    FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR    FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

_$

Va
--
00
00
00


BBBBBBBB CCCCCCCC DDDDDDDD IIIIII SSSSSSSS PPPPPPPP
BBBBBBBB CCCCCCCC DDDDDDDD IIIIII SSSSSSSS PPPPPPPP
BB BB CC DD DD II SS PP PP
BB BB CC DD DD II SS PP PP
BB BB CC DD DD II SS PP PP
BB BB CC DD DD II SS PP PP
BBBBBBBB CC DD DD II SSSSSS PPPPPPPP
BBBBBBBB CC DD DD II SSSSSS PPPPPPPP
BB BB CC DD DD II SS PP
BB BB CC DD DD II SS PP
BB BB CC DD DD II SS PP
BB BB CC DD DD II SS PP
BBBBBBBB CCCCCCCC DDDDDDDD IIIIII SSSSSSSS PP
BBBBBBBB CCCCCCCC DDDDDDDD IIIIII SSSSSSSS PP
LL IIIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

```
0001            SUBROUTINE BRIEF_C_DISPATCHER (Lun, Report_type, Record_length,
0002            1 Summary_qual, summary_flag)
0003    C
0004    C Version:      'V04-000'
0005    C
0006    C*****************************************************************************
0007    C*                                                                           *
0008    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0009    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0010    C*  ALL RIGHTS RESERVED.                                                     *
0011    C*                                                                           *
0012    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0013    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0014    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0015    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0016    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0017    C*  TRANSFERRED.                                                             *
0018    C*                                                                           *
0019    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0020    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0021    C*  CORPORATION.                                                             *
0022    C*                                                                           *
0023    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0024    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0025    C*                                                                           *
0026    C*                                                                           *
0027    C*****************************************************************************
0028    C
0029    C
0030    C
0031    C       AUTHOR: Elliott A. Drayton              CREATION DATE: 15-Dec-1982
0032    C
0033    C       Functional description:
0034    C
0035    C       This module displays brief reports for error log entries.
0036    C
0037    C       Modified by:
0038    C
0039    C       V03-014 SAR0279         Sharon A. Reynolds      26-jun-1984
0040    C               - Added TMSCP support.
0041    C               - Fixed a bug in the calling parameters to
0042    C               brief_pudriver_mscp_dispatcher.
0043    C
0044    C       V03-013 SAR0253         Sharon A. Reynolds      23-Apr-1984
0045    C               Added a field length for the output of the device
0046    C               name.
0047    C
0048    C       V03-012 SAR0251         Sharon A. Reynolds      11-Apr-1984
0049    C               Removed erf_unkentry and erf_unkclass messages. They
0050    C               are no longer needed due to the way unknown entries
0051    C               are now handled.
0052    C
0053    C       V03-011 SAR0213         Sharon A. Reynolds      28-Mar-1984
0054    C               Added KMS3271 support.
0055    C
0056    C       V03-010 EAD0123         Elliott A. Drayton              23-Mar-1984
0057    C               Changed PA logmessage text for subtypes 7 and 8.
```

```
0058    C
0059    C       V03-009 EAD0116         Elliott A. Drayton      21-Mar-1984
0060    C               Added code to report new HSC logmessage.
0061    C
0062    C       V03-008 SAR0210         Sharon A. Reynolds,     10-Mar-1984
0063    C               - Corrected the check for the call to image loader for
0064    C               /summary support for briefs.
0065    C
0066    C       V03-007 SAR0208         Sharon A. Reynolds,     8-Mar-1984
0067    C               - Fixed a bug in the output of memory entries when
0068    C               there is more than one controller.
0069    C               - Changed the way the call parameters for image loader
0070    C               were passed to work with the new image loader.
0071    C
0072    C       V03-006 SAR0186         Sharon A. Reynolds,     17-Jan-1984
0073    C               - Added message type '8' to the message type
0074    C               dispatching in the logmessage routine.
0075    C               - Added support for memory and volume summaries.
0076    C               - Added last SYE update changes.
0077    C
0078    C       V03-005 SAR0173         Sharon A. Reynolds,     11-Nov-1983
0079    C               - Added brief support for 'logmscp' entries.
0080    C
0081    C       V03-004 SAR0155         Sharon A. Reynolds,     11-Oct-1983
0082    C               - Made "RH7XX" register heralds generic.
0083    C               - Applied an SYE update that modifies the text for
0084    C                 'pudriver' support.
0085    C               - Made "DW7XX" register heralds generic.
0086    C               - Made 'padriver' changes.
0087    C               - Fixed a bug in the output of the pudriver code word
0088    C                 text.
0089    C
0090    C       V03-003 SAR0141         Sharon A. Reynolds,     23-Sep-1983
0091    C               - Put in unknown entry and device class error messages.
0092    C               - Deleted unused routines.
0093    C               - Fixed carriage control for ERF.
0094    C               - Added DR11W support.
0095    C
0096    C       V03-002 SAR0109         Sharon A. Reynolds,     21-Jun-1983
0097    C               Undefined support.
0098    C
0099    C       V03-001 SAR0033         Sharon A. Reynolds,     6-Jun-1983
0100    C               Added brief/cryptic support for memory and mount/dismount
0101    C               entries. Added a few comments.
0102    C--
0103
0104            Include 'SRC$:MSGHDR.FOR /NOLIST'
0163            Include 'SRC$:DEVERR.FOR /NOLIST'
0264            Include 'SRC$:EMBSPDEF.FOR /NOLIST'
0377            Include 'SRC$:EMBLMDEF.FOR /NOLIST'
0446            Include 'SRC$:SYECOM.FOR /NOLIST'
0574            Include 'SRC$:DEV_CLASS.FOR /NOLIST'
0616            Include 'src$:volmount.for /nolist'
0680            Include 'SRC$:SSMESSAGE.FOR /nolist'
0723            Include 'SRC$:PARSERDAT.FOR /nolist'
0965
0966            External        erf_loaderr
```

```
0967
0968            Character*1     Report_type
0969
0970            Byte            Brief_P2,       ! Number of registers to output
0971            1               Cryp_P2,        !
0972            1               Brief_32        ! 1 = 32 bit reg's, 0 = 16 bit reg's
0973
0974            Byte            Lun,            ! Logical unit number for output
0975            1               Tof,            ! Flag for call to FACTOF
0976            1               Exit1,          ! Flag to determine if to exit routine
0977            1               Write1,         ! Flag to determine if to use write
0978            1               Device_type     ! Value for device type
0979            1               Device_class    ! Value for device class
0980
0981            Data            exit1/0/,tof/0/,write1/0/
0982
0983            Byte            Summary_qual    ! Summary report indicator
0984
0985            Integer*4       Brief_P3,       ! Address of registers
0986            1               Brief_P4(2),    ! Parameters for subroutines
0987            1               Dhead_P2(2),    ! Address of id string
0988            1               Linchk_P2
0989
0990            Integer*4       Record_length,  ! Length of error packet being processed
0991            1               Compress4,      ! External function name
0992            1               Cpu_type,       ! Value for cpu type
0993            1               Reg_copy(0:11), ! General area for register values
0994            1               Length,         ! Length of string in MESSAGE
0995            1               Xfer_addr,      ! Transfer address for loadable image
0996            1               Status,         ! Return status
0997            1               Memory_summ,    ! Memory summary indicator
0998            1               All_summ,       ! Complete summary indicator
0999            1               Summary_flag    ! Summary type indicators
1000
1001            Character*80    Message
1002
1003            Character*23    image_name
1004            Data            image_name      /'SYS$SYSTEM:ERFPROC3.EXE'/
1005
1006    C
1007    C RK06 and 07 definitions
1008    C
1009    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1010    C
1011            CHARACTER*4     RK_REG_HERALD(0:11)
1012            DATA    RK_REG_HERALD(0)        /'CS1*'/
1013            DATA    RK_REG_HERALD(1)        /'WC*'/
1014            DATA    RK_REG_HERALD(2)        /'BA*'/
1015            DATA    RK_REG_HERALD(3)        /'DA*'/
1016            DATA    RK_REG_HERALD(4)        /'CS2*'/
1017            DATA    RK_REG_HERALD(5)        /'DS*'/
1018            DATA    RK_REG_HERALD(6)        /'ER*'/
1019            DATA    RK_REG_HERALD(7)        /'AS*'/
1020            DATA    RK_REG_HERALD(8)        /'DC*'/
1021            DATA    RK_REG_HERALD(9)        /'MR*'/
1022            DATA    RK_REG_HERALD(10)       /'EC1*'/
1023            DATA    RK_REG_HERALD(11)       /'EC2*'/
```

```
1024
1025
1026     C
1027     C RP device type definitions
1028     C
1029     C       Define device types and register mnemonics for the BRIEF
1030     C       and CRYPTK reports
1031     C
1032             PARAMETER       RM03 = 6
1033             PARAMETER       RP06 = 5
1034             PARAMETER       RP05 = 4
1035             PARAMETER       RP04 = 3
1036             PARAMETER       RP07 = 7
1037             PARAMETER       RM80 = 13
1038             PARAMETER       RM05 = 15
1039
1040             CHARACTER*4     REG_HERALD1(0:11)
1041             DATA    REG_HERALD1(0)  /'CS*'/
1042             DATA    REG_HERALD1(1)  /'DS*'/
1043             DATA    REG_HERALD1(2)  /'ER1*'/
1044             DATA    REG_HERALD1(3)  /'MR*'/
1045             DATA    REG_HERALD1(4)  /'AS*'/
1046             DATA    REG_HERALD1(5)  /'DA*'/
1047             DATA    REG_HERALD1(6)  /'DT*'/
1048             DATA    REG_HERALD1(7)  /'LA*'/
1049             DATA    REG_HERALD1(8)  /'SN*'/
1050             DATA    REG_HERALD1(9)  /'OF*'/
1051             DATA    REG_HERALD1(10) /'DC*'/
1052             DATA    REG_HERALD1(11) /'CC*'/
1053
1054             CHARACTER*4     REG_HERALD2(0:11)
1055             DATA    REG_HERALD2(0)  /'CS*'/
1056             DATA    REG_HERALD2(1)  /'DS*'/
1057             DATA    REG_HERALD2(2)  /'ER1*'/
1058             DATA    REG_HERALD2(3)  /'MR1*'/
1059             DATA    REG_HERALD2(4)  /'AS*'/
1060             DATA    REG_HERALD2(5)  /'DA*'/
1061             DATA    REG_HERALD2(6)  /'DT*'/
1062             DATA    REG_HERALD2(7)  /'LA*'/
1063             DATA    REG_HERALD2(8)  /'SN*'/
1064             DATA    REG_HERALD2(9)  /'OF*'/
1065             DATA    REG_HERALD2(10) /'DC*'/
1066             DATA    REG_HERALD2(11) /'HR*'/
1067
1068
1069     C
1070     C RL02 Definitions
1071     C
1072     C       Define the register mnemonics for the BRIEF and CRYPTK reports
1073     C
1074             CHARACTER*4     RL_REG_HERALD(0:3)
1075             DATA    RL_REG_HERALD(0)        /'CS*'/
1076             DATA    RL_REG_HERALD(1)        /'BA*'/
1077             DATA    RL_REG_HERALD(2)        /'DA*'/
1078             DATA    RL_REG_HERALD(3)        /'MP*'/
1079
1080
```

```
1081    C
1082    C RX01/2 Definitions
1083    C
1084    C       Define necessary data and the register mnemonics for the BRIEF
1085    C       and CRYPTK reports
1086    C
1087            Integer*4       extended_registers(2)
1088
1089            Integer*2       extended_status(4)
1090
1091            Equivalence     (extended_registers,emb$l_dv_regsav(7))
1092            Equivalence     (extended_registers,extended_status)
1093
1094            CHARACTER*5     RX_REG_HERALD(0:5)
1095            data    rx_reg_herald(0)        /'CS*'/
1096            data    rx_reg_herald(1)        /'ES*'/
1097            data    rx_reg_herald(2)        /'ES 1*'/
1098            data    rx_reg_herald(3)        /'ES 2*'/
1099            data    rx_reg_herald(4)        /'ES 3*'/
1100            data    rx_reg_herald(5)        /'ES 4*'/
1101
1102    C
1103    C TU58 Definitions
1104    C
1105    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1106    C
1107            CHARACTER*5     DD_REG_HERALD(0:3)
1108            DATA    DD_REG_HERALD(0)        /'RCSR*'/
1109            DATA    DD_REG_HERALD(1)        /'RBUF*'/
1110            DATA    DD_REG_HERALD(2)        /'XCSR*'/
1111            DATA    DD_REG_HERALD(3)        /'XBUF*'/
1112
1113
1114    C
1115    C ML11 Definitions
1116    C
1117    C
1118    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1119    C
1120            Character*5     ml_reg_herald(0:11)
1121            Data    ml_reg_herald(0)        /'CS1*'/
1122            Data    ml_reg_herald(1)        /'DS*'/
1123            Data    ml_reg_herald(2)        /'ER*'/
1124            Data    ml_reg_herald(3)        /'MR*'/
1125            Data    ml_reg_herald(4)        /'AS*'/
1126            Data    ml_reg_herald(5)        /'DA*'/
1127            Data    ml_reg_herald(6)        /'DT*'/
1128            Data    ml_reg_herald(7)        /'SN*'/
1129            Data    ml_reg_herald(8)        /'E1*'/
1130            Data    ml_reg_herald(9)        /'E2*'/
1131            Data    ml_reg_herald(10)       /'EE*'/
1132            Data    ml_reg_herald(11)       /'EL*'/
1133
1134
1135    C
1136    C RB730 Definitions
1137    C
```

```
1138    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1139    C
1140            Character*8     rb_reg_herald(0:6)
1141            data    rb_reg_herald(0)        /'RB CSR*'/
1142            data    rb_reg_herald(1)        /'RB BAR*'/
1143            data    rb_reg_herald(2)        /'RB BCR*'/
1144            data    rb_reg_herald(3)        /'RB DAR*'/
1145            data    rb_reg_herald(4)        /'RB MPR*'/
1146            data    rb_reg_herald(5)        /'RB ECC1*'/
1147            data    rb_reg_herald(6)        /'RB ECC2*'/
1148
1149
1150    C
1151    C TE16, TU45 and TU77 tapes definitions
1152    C
1153    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1154    C
1155            CHARACTER*4     tape_reg_herald(0:9)
1156            DATA    tape_reg_herald(0)      /'CS*'/
1157            DATA    tape_reg_herald(1)      /'DS*'/
1158            DATA    tape_reg_herald(2)      /'ER*'/
1159            DATA    tape_reg_herald(3)      /'MR*'/
1160            DATA    tape_reg_HERALD(4)      /'AS*'/
1161            DATA    tape_reg_HERALD(5)      /'FC*'/
1162            DATA    tape_reg_HERALD(6)      /'DT*'/
1163            DATA    tape_reg_HERALD(7)      /'CK*'/
1164            DATA    tape_reg_HERALD(8)      /'SN*'/
1165            DATA    tape_reg_HERALD(9)      /'TC*'/
1166
1167
1168    C
1169    C TS11 Definitions
1170    C
1171    C       Define necessary data and the register mnemonics for the
1172    C       BRIEF and CRYPTK reports
1173    C
1174            Character*6     ts_reg_herald(0:6)
1175            data    ts_reg_herald(0)        /'BA*'/
1176            data    ts_reg_herald(1)        /'SR*'/
1177            data    ts_reg_herald(2)        /'X0*'/
1178            data    ts_reg_herald(3)        /'X1*'/
1179            data    ts_reg_herald(4)        /'X2*'/
1180            data    ts_reg_herald(5)        /'X3*'/
1181            data    ts_reg_herald(6)        /'RBPCR*'/
1182
1183            EQUIVALENCE     (EMB$L_DV_REGSAV(15),MSG_BUF(0))
1184
1185            INTEGER*4       MSG_BUF(0:6)
1186
1187    C
1188    C TU78 Tape definitions
1189    C
1190    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1191    C
1192            character*6     mf_reg_herald(0:16)
1193            data    mf_reg_herald(0)                /'CS1*'/
1194            data    mf_reg_herald(1)                /'IS*'/
```

```
1195            data    mf_reg_herald(2)                /'TC*'/
1196            data    mf_reg_herald(3)                /'MR1*'/
1197            data    mf_reg_herald(4)                /'AB*'/
1198            data    mf_reg_herald(5)                /'BC*'/
1199            data    mf_reg_herald(6)                /'DT*'/
1200            data    mf_reg_herald(7)                /'DS*'/
1201            data    mf_reg_herald(8)                /'SN*'/
1202            data    mf_reg_herald(9)                /'MR2*'/
1203            data    mf_reg_herald(10)               /'MR3*'/
1204            data    mf_reg_herald(11)               /'NDTA*'/
1205            data    mf_reg_herald(12)               /'NDT 0*'/
1206            data    mf_reg_herald(13)               /'NDT 1*'/
1207            data    mf_reg_herald(14)               /'NDT 2*'/
1208            data    mf_reg_herald(15)               /'NDT 3*'/
1209            data    mf_reg_herald(16)               /'ID*'/
1210
1211
1212    C
1213    C DUP3271
1214    C
1215    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1216    C
1217            Character*5     dup_reg_herald(0:4)
1218            Data    dup_reg_herald(0)       /'PCSR*'/
1219            Data    dup_reg_herald(1)       /'RCSR*'/
1220            Data    dup_reg_herald(2)       /'RBUF*'/
1221            Data    dup_reg_herald(3)       /'TCSR*'/
1222            Data    dup_reg_herald(4)       /'TBUF*'/
1223
1224
1225    C
1226    C SB11
1227    C
1228    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1229    C
1230            Character*7     sb_reg_herald(0:3)
1231            Data    sb_reg_herald(0)        /'CSR0*'/
1232            Data    sb_reg_herald(1)        /'CSR1*'/
1233            Data    sb_reg_herald(2)        /'CSR2*'/
1234            Data    sb_reg_herald(3)        /'ERRSTS*'/
1235
1236    C
1237    C KMS3271
1238    C
1239    C       Define the register mnemonics for the BRIEF and CRYPTK reports
1240    C
1241            Character*5     kms3271_reg_herald(0:9)
1242            Data    kms3271_reg_herald(0)   /'DSTS*'/
1243            Data    kms3271_reg_herald(1)   /'LSTA*'/
1244            Data    kms3271_reg_herald(2)   /'CSR0*'/
1245            Data    kms3271_reg_herald(3)   /'CSR1*'/
1246            Data    kms3271_reg_herald(4)   /'CSR2*'/
1247            Data    kms3271_reg_herald(5)   /'CSR3*'/
1248            Data    kms3271_reg_herald(6)   /'CSR0*'/
1249            Data    kms3271_reg_herald(7)   /'CSR1*'/
1250            Data    kms3271_reg_herald(8)   /'CSR2*'/
1251            Data    kms3271_reg_herald(9)   /'CSR3*'/
```

```
1252
1253
1254     C
1255     C LPA11K
1256     C
1257     C       Define the register mnemonics for the BRIEF and CRYPTK reports
1258     C
1259             CHARACTER*4     lpa_reg_HERALD(0:3)
1260             DATA    lpa_reg_HERALD(0)       /'CS1*'/
1261             DATA    lpa_reg_HERALD(1)       /'CS2*'/
1262             DATA    lpa_reg_HERALD(2)       /'RDA*'/
1263             DATA    lpa_reg_HERALD(3)       /'MR*'/
1264
1265
1266     C
1267     C DR780
1268     C
1269     C       Define necessary data and the register mnemonics for the
1270     C       BRIEF and CRYPTK reports
1271     C
1272             Character*4     dr780_reg_herald (0:5)
1273             DATA    dr780_reg_HERALD(0)     /'CR*'/
1274             DATA    dr780_reg_HERALD(1)     /'UTL*'/
1275             DATA    dr780_reg_HERALD(2)     /'BA*'/
1276             DATA    dr780_reg_HERALD(3)     /'SL*'/
1277             DATA    dr780_reg_HERALD(4)     /'CA1*'/
1278             DATA    dr780_reg_HERALD(5)     /'CA2*'/
1279
1280
1281     C
1282     C dr750
1283     C
1284     C       Define the register mnemonics for the BRIEF and CRYPTK reports
1285     C
1286             Character*8     dr750_reg_herald(0:5)
1287             Data    dr750_reg_herald(0)     /'DCR*'/
1288             Data    dr750_reg_herald(1)     /'UTL*'/
1289             Data    dr750_reg_herald(2)     /'BA*'/
1290             Data    dr750_reg_herald(3)     /'WCS ADR*'/
1291             Data    dr750_reg_herald(4)     /'WCS DAT*'/
1292             Data    dr750_reg_herald(5)     /'DSL*'/
1293
1294     C
1295     C Dr11w
1296     C
1297     C       Define necessary data and the register mnemonics for the
1298     C       BRIEF and CRYPTK reports
1299     C
1300             CHARACTER*4     dr11w_reg_herald(0:4)
1301             Data    dr11w_reg_herald(0)     /'CSR*'/
1302             Data    dr11w_reg_herald(1)     /'EIR*'/
1303             Data    dr11w_reg_herald(2)     /'IDR*'/
1304             Data    dr11w_reg_herald(3)     /'BA*'/
1305             Data    dr11w_reg_herald(4)     /'WC*'/
1306
1307
1308     C
```

```
1309    C Pcl11r
1310    C
1311    C       Define necessary data and the register mnemonics for the
1312    C       BRIEF and CRYPTK reports
1313    C
1314            character*6     pcl11r_reg_herald(0:5)
1315            data            pcl11r_reg_herald(0)    /'RCR*'/
1316            data            pcl11r_reg_herald(1)    /'RSR*'/
1317            data            pcl11r_reg_herald(2)    /'RDDB*'/
1318            data            pcl11r_reg_herald(3)    /'RDBC*'/
1319            data            pcl11r_reg_herald(4)    /'RDBA*'/
1320            data            pcl11r_reg_herald(5)    /'RDCRC*'/
1321
1322
1323    C
1324    C Pcl11t
1325    C
1326    C       Define necessary data and the register mnemonics for the BRIEF
1327    C       and CRYPTK reports
1328    C
1329            character*6     pcl11t_reg_herald(0:6)
1330            data            pcl11t_reg_herald(0)    /'TCR*'/
1331            data            pcl11t_reg_herald(1)    /'TSR*'/
1332            data            pcl11t_reg_herald(2)    /'TSDB*'/
1333            data            pcl11t_reg_herald(3)    /'TSBC*'/
1334            data            pcl11t_reg_herald(4)    /'TSBA*'/
1335            data            pcl11t_reg_herald(5)    /'TMMR*'/
1336            data            pcl11t_reg_herald(6)    /'TSCRC*'/
1337
1338
1339    C
1340    C DT07
1341    C
1342    C       Define necessary data and the register mnemonics for the
1343    C       BRIEF and CRYPTK reports
1344    C
1345            integer*4       ucb$l_bs_cur
1346            integer*4       ucb$b_bs_errmsg
1347
1348            equivalence     (emb$l_dv_regsav(2),ucb$l_bs_cur)
1349            equivalence     (emb$l_dv_regsav(0),ucb$b_bs_errmsg)
1350
1351            character*5     dt07_reg_herald(0:1)
1352            data    dt07_reg_herald(0)      /'CUR*'/
1353            data    dt07_reg_herald(1)      /'PREV*'/
1354
1355
1356    C
1357    C CI780/CI750 data definitions
1358    C
1359    C       Define necessary data and the register mnemonics for the
1360    C       BRIEF and CRYPTK reports
1361    C
1362            integer*4       padriver_error_type_code
1363            integer*4       pcnfgr
1364
1365            equivalence     (emb$l_dv_regsav(0),padriver_error_type_code)
```

```
1366            equivalence     (emb$l_dv_regsav(1),pcnfgr)
1367
1368            character*6     ci_reg_herald(0:5)
1369            data    ci_reg_herald(0)        /'CNFGR*'/
1370            data    ci_reg_herald(1)        /'PMCSR*'/
1371            data    ci_reg_herald(2)        /'PSR*'/
1372            data    ci_reg_herald(3)        /'PFAR*'/
1373            data    ci_reg_herald(4)        /'PESR*'/
1374            data    ci_reg_herald(5)        /'PPR*'/
1375
1376    C
1377    C UDA50
1378    C
1379    C       Define data for the BRIEF and CRYPTK reports
1380    C
1381            integer*2       initialization_handshake(8)
1382
1383            equivalence     (emb(94),initialization_handshake)
1384
1385    C
1386    C UBA ERRORS/INTERRUPTS
1387    C
1388    C       Define necessary data and the register mnemonics for the
1389    C       BRIEF and CRYPTK reports
1390    C
1391            INTEGER*4       UBA_CSR, UBA_CR, UBA_SR, UBA_DCR, UBA_FMER
1392            INTEGER*4       UBA_FUBAR, ADAPTER_TR
1393
1394            EQUIVALENCE     (EMB(16),UBA_CSR)
1395            EQUIVALENCE     (EMB(20),UBA_CR)
1396            EQUIVALENCE     (EMB(24),UBA_SR)
1397            EQUIVALENCE     (EMB(28),UBA_DCR)
1398            EQUIVALENCE     (EMB(32),UBA_FMER)
1399            EQUIVALENCE     (EMB(36),UBA_FUBAR)
1400            EQUIVALENCE     (EMB(40),ADAPTER_TR)
1401
1402            CHARACTER*3     TR
1403            DATA    TR              /'TR '/
1404
1405            CHARACTER*6     UBA_REG_HERALD(0:5)
1406            DATA    UBA_REG_HERALD(0)       /'CSR*'/
1407            DATA    UBA_REG_HERALD(1)       /'CR*'/
1408            DATA    UBA_REG_HERALD(2)       /'SR*'/
1409            DATA    UBA_REG_HERALD(3)       /'DCR*'/
1410            DATA    UBA_REG_HERALD(4)       /'FMER*'/
1411            DATA    UBA_REG_HERALD(5)       /'FUBAR*'/
1412
1413    C
1414    C MBA interrupts
1415    C
1416    C       Define necessary data and the register mnemonics for the
1417    C       BRIEF and CRYPTK reports
1418    C
1419            integer*4       rh780_control_register(6)
1420
1421            equivalence     (emb(16),rh780_control_register)
1422
```

```
1423            character*4     mba_reg_herald(6)
1424            data    mba_reg_herald(1)       /'CSR*'/
1425            data    mba_reg_herald(2)       /'CR*'/
1426            data    mba_reg_herald(3)       /'SR*'/
1427            data    mba_reg_herald(4,       /'VAR*'/
1428            data    mba_reg_herald(5)       /'BCR*'/
1429            data    mba_reg_herald(6)       /'DR*'/
1430
1431
1432    C
1433    C BUGCHK
1434    C
1435    C       Define necessary data and the register mnemonics for the
1436    C       BRIEF and CRYPTK reports
1437    C
1438            Include 'src$:modes.for /nolist'
1476
1477            Integer         i, j, k
1478
1479            Integer*4       bugmsg_length
1480
1481            Character*73    bugmsg_text
1482            Parameter       bugchk_code = 104
1483
1484
1485    C
1486    C PUDRIVER
1487    C
1488            Integer*2       code_word
1489
1490            Equivalence     (emb(82),code_word)
1491
1492            Character*(*)   Msg1, msg2, Msg3, Atten
1493            Parameter       (
1494            1 Msg1 = '"AVAILABLE" ',
1495            2 Msg2 = '"DUPLICATE UNIT NUMBER" ',
1496            3 Msg3 = '"ACCESS PATH" ',
1497            4 Atten = 'MSCP ATTENTION MESSAGE')
1498
1499            Character*(*)   Init, Comp, Fail, Bus_msg1, Bus_msg2, Bus_msg3
1500            Parameter       (
1501            1 Init = ', INIT SEQUENCE ',
1502            2 Comp = 'COMPLETED',
1503            3 Fail = 'FAILURE'
1504            4 Bus_msg1 = ', "SA" ERROR BIT SET',
1505            5 Bus_msg2 = ', UBA DATAPATH PURGE ERROR',
1506            6 Bus_msg3 = ', UCODE REV AND "PUDRIVER" MIS-MATCH')
1507
1508
1509    C Logmscp (mscp messages without ucb's)
1510    C
1511            Integer*2       logmscp_type
1512            Equivalence     (emb(16),logmscp_type)
1513
1514
1515
1516    C Common memory entry data definitions
```

```
1517    C
1518            integer*4       memory_registers
1519            integer*4       buffer(0:120)
1520            integer*4       mem_reg_buffer(8)
1521            integer*4       memory_register_0
1522            integer*4       memory_register_a
1523            integer*4       controller_count
1524
1525            equivalence     (mem_reg_buffer(1),memory_register_a)
1526            equivalence     (emb(16),buffer(0))
1527            equivalence     (buffer(0),controller_count)
1528            equivalence     (buffer(2),memory_registers)
1529
1530            character*5     mem_reg_herald(3)
1531            data            mem_reg_herald(1)/'CSR0*'/
1532            data            mem_reg_herald(2)/'CSR1*'/
1533            data            mem_reg_herald(3)/'CSR2*'/
1534
1535    C  Ma780
1536    C
1537    C       Define necessary data and the register mnemonics for the
1538    C       BRIEF and CRYPTK reports
1539    C
1540            integer*4       port_configuration_register
1541
1542            equivalence     (mem_reg_buffer(1),port_configuration_register)
1543
1544            character*8     ma780_reg_herald(8)
1545            data            ma780_reg_herald(1)/'PRTCFNG*'/
1546            data            ma780_reg_herald(2)/'PRTCR*'/
1547            data            ma780_reg_herald(3)/'PCSR*'/
1548            data            ma780_reg_herald(4)/'IVDTCR*'/
1549            data            ma780_reg_herald(5)/'AER*'/
1550            data            ma780_reg_herald(6)/'CSR0*'/
1551            data            ma780_reg_herald(7)/'CSR1*'/
1552            data            ma780_reg_herald(8)/'MAT*'/
1553
1554    C
1555    C  MS780C
1556    C
1557    C       Define necessary data and the register mnemonics for the
1558    C       BRIEF and CRYPTK reports
1559    C
1560            character*5     ms780c_reg_herald(3)
1561            data            ms780c_reg_herald(1)/'CSRA*'/
1562            data            ms780c_reg_herald(2)/'CSRB*'/
1563            data            ms780c_reg_herald(3)/'CSRC*'/
1564
1565    C
1566    C  MS780E
1567    C
1568    C       Define necessary data and the register mnemonics for the
1569    C       BRIEF and CRYPTK reports
1570    C
1571            character*5     ms780e_reg_herald(4)
1572            data            ms780e_reg_herald(1)/'CSRA*'/
1573            data            ms780e_reg_herald(2)/'CSRB*'/
```

```
1574            data            ms780e_reg_herald(3)/'CSRC*'/
1575            data            ms780e_reg_herald(4)/'CSRD*'/
1576
1577    C
1578    C  MS750
1579    C
1580    C       Define necessary data and the register mnemonics for the
1581    C       BRIEF and CRYPTK reports
1582    C
1583            integer*4       slot_index
1584
1585            equivalence     (mem_reg_buffer(1),memory_register_0)
1586
1587    C
1588    C  MS730
1589    C
1590    C       Define necessary data and the register mnemonics for the
1591    C       BRIEF and CRYPTK reports
1592    C
1593            integer*4       memory_csr0
1594
1595            equivalence     (mem_reg_buffer(1),memory_csr0)
1596
1597    C
1598    C MOUNT/DISMOUNT data definitions
1599    C
1600    C Define the necessary data and the register mnemonics for the
1601    C BRIEF/CRYPTIC reports
1602    C
1603            logical*1       str$trim
1604
1605            integer*4       emb$t_vm_label_length
1606            integer*4       mount_operation_count
1607            integer*4       mount_error_count
1608            integer*4       volume_operation_count
1609            integer*4       volume_error_count
1610
```

```
1611    C
1612    C Determine if a summary report should be output. If so, load
1613    C SYS$SYSTEM:ERFPROC3.EXE for volume and memory summaries.
1614    C
1615            If (summary_qual .EQ. 1) then
1616
1617              All_summ = LIB$EXTZV(sum$v_all_summ,sum$s_all_summ,summary_flag)
1618              Memory_summ = LIB$EXTZV(sum$v_memory,sum$s_memory,summary_flag)
1619
1620              If ( ((all_summ)
1621            1 .OR. (memory_summ))
1622            1 .AND. (emb$w_hd_entry .EQ. 6          ! EMB$K_SE
1623            1 .OR. emb$w_hd_entry .EQ. 8            ! EMB$K_HE
1624            1 ) ) then
1625
1626                Status = IMAGE_LOADER (emb$w_hd_entry,device_class,xfer_addr)
1627
1628                If (xfer_addr .EQ. 0) then
1629                  Call LIB$SIGNAL (status)
1630    !?? Should I append a message indicating that memory summaries cannot be done.
1631                Endif
1632              Endif
1633            Endif
1634
1635
1636    C
1637    C Via the entry type, dispatch to the appropriate entry handling routine.
1638    C
1639            Goto (1,2,999,4,5,6,7,8,9,999) emb$w_hd_entry
1640            Goto (11, 12 ) (emb$w_hd_entry - 10)
1641            Goto (999,32,999,999,35,36,37,38,39,40 ) (emb$w_hd_entry - 30)
1642            Goto (41, 42 ) (emb$w_hd_entry - 40)
1643            Goto (999,999,999,64,65) (emb$w_hd_entry - 60)
1644            Goto (999,999,999,999,999,96,97,98,99,100 ) (emb$w_hd_entry - 90)
1645            Goto (101) (emb$w_hd_entry - 100)
1646            Goto (999,112) (emb$w_hd_entry - 110)
1647
1648            Return
1649
1650
1651    C DEVICE ERROR, TIMEOUT, and ATTENTION entries
1652    C
1653    C Via device class, dispatch to the appropriate device dispatching routine.
1654    C
1655    1       Continue          ! Device error packet
1656    96      Continue          ! Device timeout packet
1657    98      Continue          ! Device attention packet
1658
1659               Device_type = lib$extzv (0,8,emb$b_dv_type)
1660
1661               If ( EMB$B_DV_CLASS .EQ. DC$_DISK ) goto 10000
1662               If ( EMB$B_DV_CLASS .EQ. DC$_TAPE ) goto 20000
1663               If ( EMB$B_DV_CLASS .EQ. DC$_SCOM ) goto 30000
1664               If ( EMB$B_DV_CLASS .EQ. DC$_REALTIME ) goto 40000
1665               If ( EMB$B_DV_CLASS .EQ. DC$_BUS ) goto 50000
1666
1667            Return
```

```
1668    C DISKS device dispatching routine.
1669    C
1670    C Via device type dispatch to the appropriate device dependent routine.
1671    C
1672    10000   goto (
1673
1674             1 10001,               ! - rk06
1675             1 10002,               ! - rk07
1676
1677             1 10003,               ! - rp04
1678             1 10004,               ! - rp05
1679
1680             1 10005,               ! - rp06
1681             1 10006,               ! - rm03
1682
1683             1 10007,               ! - rp07
1684             1 999,                 ! - return
1685
1686             1 10009,               ! - rl01
1687             1 10010,               ! - rl02
1688
1689             1 10011,               ! - rx02
1690             1 10012,               ! - rx03
1691
1692             1 10013,               ! - rm80
1693             1 10014,               ! - tu58
1694
1695             1 10015,               ! - rm05
1696             1 999,                 ! - rx01
1697
1698             1 10017,               ! - ml11
1699             1 10018,               ! - rb02
1700
1701             1 10019                ! - rb80
1702                                    ! The ra80, ra81, ra60,
1703                                    ! rc25, and rcf25 are logged
1704                                    ! as logmessage entries and will
1705                                    ! be output by that brief code.
1706             1 ) Device_type
1707
1708             Return
1709
1710
```

```
1711    C RK611 device dependent routine
1712    C
1713    C Set up the brief and cryptic call parameters and go call the
1714    C brief/cryptic routines.
1715    C
1716    10001   continue
1717
1718    10002   Brief_P2 = 12
1719            Brief_P3 = %loc(EMB$L_DV_REGSAV(0))
1720            Brief_P4(1) = 4
1721            Brief_P4(2) = %loc(rk_reg_herald)
1722            Cryp_P2 = 16
1723            Dhead_P2(1) = 9
1724            Dhead_P2(2) = %loc ('UBA RK611')
1725
1726            Goto 1000
1727
1728
1729    C RP 4,5 and 6 device dependent routine
1730    C
1731    C Set up the brief and cryptic call parameters and go call the
1732    C brief/cryptic routines.
1733    C
1734    10003   continue
1735    10004   continue
1736    10005   continue
1737
1738    10007   Brief_P2 = 12
1739            Brief_P3 = %loc(EMB$L_DV_REGSAV(7))     ! Drive_CS1
1740            Brief_P4(1) = 4
1741            Brief_P4(2) = %loc(reg_herald1)
1742            Cryp_P2 = 16
1743            Dhead_P2(1) = 7
1744            Dhead_P2(2) = %loc ('MASSBUS')
1745
1746            Goto 1000
1747
1748
1749    C RM 03,05 and 80 device dependent routine
1750    C
1751    C Set up the brief and cryptic call parameters and go call the
1752    C brief/cryptic routines.
1753    C
1754    10006   continue
1755    10013   continue
1756
1757    10015   Brief_P2 = 12
1758            Brief_P3 = %loc(EMB$L_DV_REGSAV(7))     ! Drive_CS1
1759            Brief_P4(1) = 4
1760            Brief_P4(2) = %loc(reg_herald2)
1761            Cryp_P2 = 16
1762            Dhead_P2(1) = 7
1763            Dhead_P2(2) = %loc ('MASSBUS')
1764
1765            Goto 1000
1766
```

```
C RL 01 and 02 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10009   continue

10010   Brief_P2 = 4
        Brief_P3 = %loc(EMB$L_DV_REGSAV(0))
        Brief_P4(1) = 4
        Brief_P4(2) = %loc(rl_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 8
        Dhead_P2(2) = %loc ('UBA RL11')

        Goto 1000


C RX02 device dependent routine
C
C Set up the brief and cryptic call parameters and go call the
C brief/cryptic routines.
C
10011   Continue

10012   Reg_copy(1) = EMB$L_DV_REGSAV(0)        ! rx2_cs
        Reg_copy(2) = EMB$L_DV_REGSAV(1)        ! rx2_es
        Reg_copy(3) = lib$extzv(0,16,extended_registers(1))
        Reg_copy(4) = lib$extzv(16,16,extended_registers(1))
        Reg_copy(5) = lib$extzv(0,16,extended_registers(2))
        Reg_copy(6) = lib$extzv(16,16,extended_registers(2))

        Brief_P2 = 6
        Brief_P3 = %loc(Reg_copy)
        Brief_P4(1) = 5
        Brief_P4(2) = %loc(rx_reg_herald)
        Cryp_P2 = 16
        Dhead_P2(1) = 9
        Dhead_P2(2) = %loc ('UBA RX211')

        Goto 1000

```

```
1809    C TU58 device dependent routine
1810    C
1811    C Set up the brief and cryptic call parameters and go call the
1812    C brief/cryptic routines.
1813    C
1814    10014   Brief_P2 = 4
1815            Brief_P3 = %loc(EMB$L_DV_REGSAV(2))     ! RCSR
1816            Brief_P4(1) = 5
1817            Brief_P4(2) = %loc(dd_reg_herald)
1818            Cryp_P2 = 16
1819
1820    C
1821    C Determine whether there is a node name logged.
1822    C
1823    C Search device name string for a '$'.
1824    C
1825            I = LIB$LOCC (emb$t_dv_name,'$')
1826
1827    C
1828    C If a node name was logged account for and determine whether this
1829    C tu58 entry is for the console tu58.
1830    C
1831            If (I .NE. 0) then
1832                If (emb$t_dv_name((I+1):3) .eq. 'CSA') then J = 1
1833            Else
1834                If (emb$t_dv_name(1:3) .eq. 'CSA') then J = 1
1835            Endif
1836
1837    C
1838    C Indicate whether this entry was for a console tu58 or uba tu58.
1839    C
1840            If (J .EQ. 1) then
1841                Dhead_P2(1) = 12
1842                Dhead_P2(2) = %loc ('CONSOLE TU58')
1843            Else
1844                Dhead_P2(1) = 8
1845                Dhead_P2(2) = %loc ('UBA TU58')
1846            Endif
1847
1848            Goto 1000
1849
1850
1851    C ML11 device dependent routine
1852    C
1853    C Make a sequential copy of the device registers. Set up the brief
1854    C and cryptic call parameters and go call the brief/cryptic routines.
1855    C
1856    10017   Reg_copy(0) = EMB$L_DV_REGSAV(8)         ! MLCS1
1857            Reg_copy(1) = EMB$L_DV_REGSAV(9)         ! MLDS
1858            Reg_copy(2) = EMB$L_DV_REGSAV(10)        ! MLER
1859            Reg_copy(3) = EMB$L_DV_REGSAV(11)        ! MLMR
1860            Reg_copy(4) = EMB$L_DV_REGSAV(12)        ! MLAS
1861            Reg_copy(5) = EMB$L_DV_REGSAV(13)        ! MLDA
1862            Reg_copy(6) = EMB$L_DV_REGSAV(14)        ! MLDT
1863            Reg_copy(7) = EMB$L_DV_REGSAV(16)        ! MLSN
1864            Reg_copy(8) = EMB$L_DV_REGSAV(17)        ! MLE1
1865            Reg_copy(9) = EMB$L_DV_REGSAV(18)        ! MLE2
```

```
1866            Reg_copy(10) = EMB$L_DV_REGSAV(21)      ! MLEE
1867            Reg_copy(11) = EMB$L_DV_REGSAV(22)      ! MLEL
1868
1869            Brief_P2 = 12
1870            Brief_P3 = %loc(Reg_copy)
1871            Brief_P4(1) = 5
1872            Brief_P4(2) = %loc(ml_reg_herald)
1873            Cryp_P2 = 16
1874            Dhead_P2(1) = 7
1875            Dhead_P2(2) = %loc ('MASSBUS')
1876
1877            Goto 1000
1878
```

```
1879    C RB 02 and 80 device dependent routine
1880    C
1881    C Set up the brief and cryptic call parameters and go call the
1882    C brief/cryptic routines.
1883    C
1884    10018   continue
1885
1886    10019   Brief_P2 = 7
1887            Brief_P3 = %loc(emb$l_dv_regsav(0))     ! Control and status register
1888            Brief_P4(1) = 8
1889            Brief_P4(2) = %loc(rb_reg_herald)
1890            Cryp_P2 = 32
1891            Dhead_P2(1) = 5
1892            Dhead_P2(2) = %loc ('RB730')
1893            Brief_32 = 1
1894
1895            Goto 1000
1896
1897
```

```
1898    C TAPES device dispatching routine.
1899    C
1900    C Via device type dispatch to the appropriate device dependent routine.
1901    C
1902    20000   continue
1903
1904            goto (
1905            1 201,          ! - te16
1906            1 202,          ! - tu45
1907            1 203,          ! - tu77
1908            1 204,          ! - ts11
1909            1 205           ! - tu78
1910            1 ) Device_type
1911
1912            Return
1913
1914
1915    C TE16, TU45, TU77 device dependent routine
1916    C
1917    C Set up the brief and cryptic call parameters and go call the
1918    C brief/cryptic routines.
1919    C
1920    201     Continue
1921    202     Continue
1922
1923    203     Brief_P2 = 10
1924            Brief_P3 = %loc(EMB$L_DV_REGSAV(7))     ! Drive_CS1
1925            Brief_P4(1) = 4
1926            Brief_P4(2) = %loc(tape_reg_herald)
1927            Cryp_P2 = 16
1928            Dhead_P2(1) = 7
1929            Dhead_P2(2) = %loc ('MASSBUS')
1930
1931            Goto 1000
1932
1933    C TS11 device dependent routine
1934    C
1935    C Set up the brief and cryptic call parameters and go call the
1936    C brief/cryptic routines.
1937    C
1938    204     Reg_copy(0) = EMB$L_DV_REGSAV(0)        ! TSBA
1939            Reg_copy(1) = EMB$L_DV_REGSAV(1)        ! TSSR
1940            Reg_copy(2) = msg_buf(3)                ! Xstat0
1941            Reg_copy(3) = msg_buf(4)                ! Xstat1
1942            Reg_copy(4) = msg_buf(5)                ! Xstat
1943            Reg_copy(5) = msg_buf(6)                ! Xstat3
1944            Reg_copy(6) = msg_buf(2)                ! Rbpcr
1945
1946            Brief_P2 = 7
1947            Brief_P3 = %loc(Reg_copy)
1948            Brief_P4(1) = 6
1949            Brief_P4(2) = %loc(ts_reg_herald)
1950            Cryp_P2 = 16
1951            Dhead_P2(1) = 8
1952            Dhead_P2(2) = %loc ('UBA TS11')
1953
1954            Goto 1000
```

```
1955
1956
1957     C TU78 device dependent routine
1958     C
1959     C Set up the brief and cryptic call parameters and go call the
1960     C brief/cryptic routines.
1961     C
1962     205      Brief_P2 = 17
1963              Brief_P3 = %loc(emb$l_dv_regsav(7))      ! MF_CS1
1964              Brief_P4(1) = 6
1965              Brief_P4(2) = %loc(mf_reg_herald)
1966              Cryp_P2 = 16
1967              Dhead_P2(1) = 7
1968              Dhead_P2(2) = %loc ('MASSBUS')
1969
1970              Goto 1000
1971
1972
```

```
1973    C
1974    C SCOM (synchronous communications) device dispatching routine.
1975    C
1976    C Via device type dispatch to the appropriate device dependent routine.
1977    C
1978    30000   continue
1979
1980            goto (
1981           1 999,           ! - use other dispatcher
1982           1 999,           ! - use other dispatcher
1983           1 303,           ! - dup11 (3271)
1984           1 999,           ! - use other dispatcher
1985           1 999,           ! - use other dispatcher
1986           1 999,           ! - use other dispatcher
1987           1 307,           ! - sb11
1988           1 999,           ! - use other dispatcher
1989           1 999,           ! - use other dispatcher
1990           1 999,           ! - use other dispatcher
1991           1 999,           ! - use other dispatcher
1992           1 999,           ! - use other dispatcher
1993           1 999,           ! - use other dispatcher
1994           1 999,           ! - use other dispatcher
1995           1 999,           ! - use other dispatcher
1996           1 999,           ! - use other dispatcher
1997           1 999,           ! - use other dispatcher
1998           1 318            ! - KMS3271
1999           1 ) Device_type
2000
2001            Return
2002
2003
2004    C DUP11/3271 device dependent routine
2005    C
2006    C Set up the brief and cryptic call parameters and go call the
2007    C brief/cryptic routines.
2008    C
2009    303     If ( Report_type .EQ. 'C' ) Return
2010            Brief_P2 = 5
2011            Brief_P3 = %loc(EMB$L_DV_REGSAV(1))     ! PARCSR
2012            Brief_P4(1) = 5
2013            Brief_P4(2) = %loc(dup_reg_herald)
2014            Dhead_P2(1) = 9
2015            Dhead_P2(2) = %loc ('UBA DUP11')
2016
2017            Goto 1000
2018
2019    C
2020    C SB11 device dependent routine
2021    C
2022    C Make a sequential copy of the device registers. Set up the brief
2023    C and cryptic call parameters and go call the brief/cryptic routines.
2024    C
2025    307     Reg_copy(0) = LIB$EXTZV (0,8,EMB$L_DV_REGSAV(1)) ! registers
2026            Reg_copy(1) = LIB$EXTZV (8,8,EMB$L_DV_REGSAV(1))
2027            Reg_copy(2) = LIB$EXTZV (16,8,EMB$L_DV_REGSAV(1))
2028            Reg_copy(3) = EMB$L_DV_REGSAV(0)                ! err_sts
2029
```

```
2030            Brief_P2 = 4
2031            Brief_P3 = %loc(reg_copy)
2032            Brief_P4(1) = 7
2033            Brief_P4(2) = %loc(sb_reg_herald)
2034            Cryp_P2 = 32
2035            Dhead_P2(1) = 15
2036            Dhead_P2(2) = %loc ('UBA DEC DATAWAY')
2037            Brief_32 = 1
2038
2039            Goto 1000
2040
2041    C
2042    C KMS3271 device dependent routine
2043    C
2044    C Set up the brief and cryptic call parameters and go call the
2045    C brief/cryptic routines.
2046    C
2047    318     Continue
2048            Brief_p2 = 10
2049            Brief_p3 = %LOC(emb$l_dv_regsav(0))
2050            Brief_p4(1) = 5
2051            Brief_p4(2) = %LOC(kms3271_reg_herald)
2052            Cryp_p2 = 16
2053            Dhead_p2(1) = 11
2054            Dhead_p2(2) = %LOC('UBA KMS3271')
2055
2056            Goto 1000
2057
```

```
2058    C
2059    C REALTIME device dispatching routine.
2060    C
2061    C Via device type dispatch to the appropriate device dependent routine.
2062    C
2063    40000   continue
2064
2065            goto (
2066            1 401,          ! - lpa11
2067            1 402,          ! - dr780
2068            1 403,          ! - dr750
2069            1 404,          ! - dr11w
2070            1 405,          ! - pcl11 (receiver)
2071            1 406,          ! - pcl11 (transmitter)
2072            1 999,          ! - spare
2073            1 408           ! - dt07
2074            1 ) Device_type
2075
2076            Return
2077
2078    C LPA11K device dependent routine
2079    C
2080    C Set up the brief and cryptic call parameters and go call the
2081    C brief/cryptic routines.
2082    C
2083    401     Brief_P2 = 4
2084            Brief_P3 = %loc(EMB$L_DV_REGSAV(0))     ! CNTRL_MAINT
2085            Brief_P4(1) = 4
2086            Brief_P4(2) = %loc(lpa_reg_herald)
2087            Cryp_P2 = 16
2088            Dhead_P2(1) = 10
2089            Dhead_P2(2) = %loc ('UBA LPA11K')
2090
2091            Goto 1000
2092
2093
2094    C DR780 device dependent routine
2095    C
2096    C Make a sequential copy of the device registers. Set up the brief
2097    C and cryptic call parameters and go call the brief/cryptic routines.
2098    C
2099    402     Reg_copy(0) = EMB$L_DV_REGSAV(0)        ! DR_CR
2100            Reg_copy(1) = EMB$L_DV_REGSAV(1)        ! DR_UTL
2101            Reg_copy(2) = EMB$L_DV_REGSAV(5)        ! DR_BA
2102            Reg_copy(3) = EMB$L_DV_REGSAV(16)       ! DR_SL
2103            Reg_copy(4) = EMB$L_DV_REGSAV(9)        ! DR_CA(1)
2104            Reg_copy(5) = EMB$L_DV_REGSAV(10)       ! DR_CA(2)
2105
2106            Brief_P2 = 6
2107            Brief_P3 = %loc(Reg_copy)
2108            Brief_P4(1) = 4
2109            Brief_P4(2) = %loc(dr780_reg_herald)
2110            Cryp_P2 = 32
2111            Dhead_P2(1) = 4
2112            Dhead_P2(2) = %loc ('DR32')
2113            Brief_32 = 1
2114
```

```
2115              Goto 1000
2116
2117
```

```
2118    C DR750 device dependent routine
2119    C
2120    C Make a sequential copy of the device registers. Set up the brief
2121    C and cryptic call parameters and go call the brief/cryptic routines.
2122    C
2123    403     Reg_copy(0) = EMB$L_DV_REGSAV(0)        ! DR_DCR
2124            Reg_copy(1) = EMB$L_DV_REGSAV(1)        ! DR_UTL
2125            Reg_copy(2) = EMB$L_DV_REGSAV(5)        ! DR_CMI_ADR
2126            Reg_copy(3) = EMB$L_DV_REGSAV(2)        ! DR_WCS_ADR
2127            Reg_copy(4) = EMB$L_DV_REGSAV(3)        ! DR_WCS_DAT
2128            Reg_copy(5) = EMB$L_DV_REGSAV(8)        ! DR_DSL
2129
2130            Brief_P2 = 6
2131            Brief_P3 = %loc(reg_copy)
2132            Brief_P4(1) = 8
2133            Brief_P4(2) = %loc(dr750_reg_herald)
2134            Cryp_P2 = 32
2135            Dhead_P2(1) = 4
2136            Dhead_P2(2) = %loc ('DR32')
2137            Brief_32 = 1
2138
2139            Goto 1000
2140
2141
2142    C DR11W device dependent routine
2143    C
2144    C Set up the brief and cryptic call parameters and go call the
2145    C brief/cryptic routines.
2146    C
2147    404     Brief_P2 = 5
2148            Brief_P3 = %LOC (emb$l_dv_regsav(0))
2149            Brief_P4(1) = 4
2150            Brief_P4(2) = %LOC (dr11w_reg_herald)
2151            Cryp_P2 = 16
2152            Dhead_P2(1) = 9
2153            Dhead_P2(2) = %LOC ('UBA DR11W')
2154
2155            Goto 1000
2156
2157
2158    C PCL11R device dependent routine
2159    C
2160    C Set up the brief and cryptic call parameters and go call the
2161    C brief/cryptic routines.
2162    C
2163    405     Brief_P2 = 6
2164            Brief_P3 = %loc(EMB$L_DV_REGSAV(0))             ! RCR
2165            Brief_P4(1) = 6
2166            Brief_P4(2) = %loc(pcl11r_reg_herald)
2167            Cryp_P2 = 16
2168            Dhead_P2(1) = 20
2169            Dhead_P2(2) = %loc ('UBA PCL11 (RECEIVER)')
2170
2171            Goto 1000
2172
2173
2174    C PCL11T device dependent routine
```

```
2175    C
2176    C Set up the brief and cryptic call parameters and go call the
2177    C brief/cryptic routines.
2178    C
2179    406     Brief_P2 = 7
2180            Brief_P3 = %loc(EMB$L_DV_REGSAV(0))             ! TCR
2181            Brief_P4(1) = 6
2182            Brief_P4(2) = %loc(pcl11t_reg_herald)
2183            Cryp_P2 = 16
2184            Dhead_P2(1) = 23
2185            Dhead_P2(2) = %loc ('UBA PCL11 (TRANSMITTER)')
2186
2187            Goto 1000
2188
2189    C UNUSED
2190    C
2191    407     Return
2192
2193
2194    C DT07 device dependent routine
2195    C
2196    C Set up the brief and cryptic call parameters and go call the
2197    C brief/cryptic routines.
2198    C
2199    408     If ( Report_type .EQ. 'B' ) then
2200
2201            call dhead1 (lun,'UBA DT07')
2202            call linchk (lun,2)
2203            call ucb$$b_bs_errmsg (lun,ucb$b_bs_errmsg)
2204
2205            call brief16 (lun,(2),ucb$l_bs_cur,dt07_reg_herald,emb$t_dv_name,
2206            1 emb$w_dv_unit)
2207
2208            Else
2209
2210            call cryptk (lun,16,(2),ucb$l_bs_cur,dt07_reg_herald,
2211            1 emb$t_dv_name,emb$w_dv_unit)
2212            Endif
2213
2214            Return
2215
2216
2217
```

```
2218    C
2219    C BUS device dispatching routine.
2220    C
2221    C Via device type dispatch to the appropriate device dependent routine.
2222    C
2223    50000   continue
2224
2225            goto (
2226           1 501,   ! - ci780
2227           1 502,   ! - ci750
2228           1 503,   ! - uda50
2229           1 504,   ! - uda50a
2230           1 999,   ! - rc25
2231           1 999    ! - tu81p
2232           1 ) Device_type
2233
2234            Return
2235
2236
2237    C CI780/CI750 device dependent routine
2238    C
2239    C Set up the brief and cryptic call parameters and go call the
2240    C brief/cryptic routines.
2241    C
2242    501     Continue
2243    502     Continue
2244
2245            If ( Report_type .EQ. 'B' ) then
2246
2247            call header (lun)
2248            call logger (lun,'DEVICE ATTENTION')
2249            call padriver_attention_error_code (lun,padriver_error_type_code)
2250
2251            call brief32 (lun,(6),pcnfgr,ci_reg_herald,
2252         1   emb$t_dv_name,emb$w_dv_unit)
2253
2254            Else
2255
2256            call cryptk (lun,32,(6),pcnfgr,ci_reg_herald,
2257         1   emb$t_dv_name,emb$w_dv_unit)
2258            Endif
2259
2260            Return
2261
2262
2263    C UDA50 device dependent routine
2264    C
2265    C Set up the brief and cryptic call parameters and go call the
2266    C brief/cryptic routines.
2267    C
2268    503     If ( Report_type .EQ. 'B' ) then
2269    504     Continue
2270
2271            call header (lun)
2272            call logger (lun,'DEVICE ATTENTION')
2273            call linchk (lun,6)
2274
```

```
2275            Goto (5001, 5002, 5003, 5004, 5005) code_word
2276
2277            write(lun,5210) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
2278            1 ', "PUDRIVER" CODE #',code_word,'.'
2279            Goto 5007
2280
2281    5001    Message = init // comp
2282            Length = len (init) + len (comp)
2283            Goto 5006
2284
2285    5002    Message = init // fail
2286            Length = len (init) + len (fail)
2287            Goto 5006
2288
2289    5003    Message = bus_msg1
2290            Length = len (bus_msg1)
2291            Goto 5006
2292
2293    5004    Message = bus_msg2
2294            Length = len (bus_msg2)
2295            Goto 5006
2296
2297    5005    Message = bus_msg3
2298            Length = len (bus_msg3)
2299
2300    5006    write(lun,5210) emb$t_dv_name(1:emb$b_dv_namlng),emb$w_dv_unit,
2301            1 Message
2302    5210    format(/' ','"DSA" PORT SUB-SYSTEM, UNIT ',a<emb$b_dv_namlng>,
2303            1 i<compress4 (lib$extzv(0,16,emb$w_dv_unit))>,':',:a,
2304            1 :i<compress4 (lib$extzv(0,16,code_word))>,:a<length>)
2305
2306    5007    write(lun,5215) 'SA','PSTEP1','HSTEP1','PSTEP2','HSTEP2','PSTEP3',
2307            1 'HSTEP3','PSTEP4','HSTEP4'
2308    5215    format(/' ',t8,a,t15,a,t22,a,t29,a,t36,a,t43,a,t50,a,t57,a,t64,a)
2309
2310            write(lun,5220) uda_sa,(initialization_handshake(i),i = 1,8)
2311    5220    format(/' ',t8,z4.4,8(' ',z4.4))
2312
2313            endif
2314
2315            Return
2316
2317
```

```
2318     C MACHINE CHECK entries
2319     C
2320     C Output the cryptic report for the above entries.
2321     C
2322     2       If ( Report_type .EQ. 'C' ) Return      ! Mach. check packet
2323
2324             Call frctof (lun)
2325             Call header (lun)
2326             Call logger (lun,'MACHINE CHECK')
2327
2328             Return
2329
2330     C SBI fault and ASYNC WRITE ERROR entries
2331     C
2332     C Output the cryptic report for the above entries.
2333     C
2334     4       Continue        ! SBI fault packet
2335     7       Continue        ! SBI async write packet
2336
2337             If ( Report_type .EQ. 'C' ) Return
2338
2339             Call header (lun)
2340
2341             Cpu_type = lib$extzv(24,8,emb$l_hd_sid)
2342
2343             If (
2344             1 cpu_type .eq. 255
2345             1 .or.
2346             1 cpu_type .eq. 1
2347             1 .OR.
2348             1 cpu_type .EQ. 7
2349             1 ) then
2350
2351             If (emb$w_hd_entry .eq. 7 ) then
2352               Call logger (lun,'ASYNCHRONOUS WRITE')
2353             Else
2354               Call logger (lun,'SBI FAULT')
2355             ENDIF
2356
2357             else if ( cpu_type .eq. 2 ) then
2358
2359             if (emb$w_hd_entry .eq. 7 ) then
2360
2361             call logger (lun,'WRITE BUS ERROR')
2362             endif
2363             endif
2364
2365             RETURN
2366
2367
2368
2369     C
2370     C UBA INTERRUPT entries
2371     C
2372     C Output the brief and cryptic reports for the above entry type.
2373     C
2374
```

```
2375    9       If ( Report_type .EQ. 'B' ) then
2376              Call header (lun)
2377              Call logger (lun,'"DW" INTERRUPT')
2378              Call brief32 (lun,(6),uba_csr,uba_reg_herald,tr,adapter_tr)
2379            Else
2380              Call cryptk (lun,32,(6),uba_csr,uba_reg_herald,tr,adapter_tr)
2381            Endif
2382            Return
2383
2384
2385    C
2386    C UBA ERROR entries
2387    C
2388    C Output the brief and cryptic reports for the above entry type.
2389    C
2390
2391    11      If ( Report_type .EQ. 'B' ) then
2392              Call header (lun)
2393              Call logger (lun,'"DW" ERROR')
2394              Call brief32 (lun,(1),uba_csr,'UBA CSR','UBA',0)
2395            Else
2396              Call cryptk (lun,32,(1),uba_csr,'UBA CSR','UBA',0)
2397            Endif
2398            Return
2399
2400
2401    C
2402    C MBA INTERRUPT entries
2403    C
2404    C Output the brief and cryptic reports for the above entry type.
2405    C
2406
2407    12      If ( Report_type .EQ. 'B' ) then
2408              Call header (lun)
2409              Call logger (lun,'"RH" INTERRUPT')
2410              Call brief32 (lun,(6),rh780_control_register,reg_herald,TR,
2411            1 adapter_tr)
2412            Else
2413              Call cryptk (lun,32,(6),rh780_control_register,reg_herald,TR,
2414            1 adapter_tr)
2415            Endif
2416            Return
2417
```

```
2418    C SYSTEM START-UP and SYSTEM POWER FAIL RECOVERY entries
2419    C
2420    C Output the brief report for the above entry types.
2421    C
2422    32      Continue        ! system start-up
2423    36      Continue        ! system power fail recovery
2424
2425            Call header (lun)
2426
2427            If (emb$w_hd_entry .eq. 32 ) then
2428              Call logger (lun,'SYSTEM START-UP')
2429            Else if (emb$w_hd_entry .eq. 36 ) then
2430              Call logger (lun,'POWER-FAIL RESTART')
2431            Endif
2432            Return
2433
2434
2435    C SYSTEM POWER FAIL entries
2436    C
2437    C Output the brief report for the above entry type.
2438    C
2439    34      Call header (lun)               ! system power fail
2440            Call logger (lun,'SYSTEM POWER-FAIL')
2441            Return
2442
2443    C NEW-FILE CREATION and TIME STAMP entries
2444    C
2445    C Output the brief report for the above entry types.
2446    C
2447    35      Continue        ! errlog.sys created
2448    38      Continue        ! time stamp
2449
2450            Call header (lun)
2451
2452            If (emb$w_hd_entry .eq. 35) then
2453              Call logger (lun,'ERRLOG.SYS CREATED')
2454            Else if (emb$w_hd_entry .eq. 38) ther
2455              Call logger (lun,'TIME STAMP')
2456            Endif
2457
2458            Return
2459
2460    C BUGCHECK entries
2461    C
2462    C Output the brief report for the above entry types.
2463    C
2464    37      Continue        ! fatal bugcheck
2465    40      Continue        ! non-fatal bugcheck
2466    112     Continue        ! user bugcheck
2467
2468    910     format(/' ',<bugmsg_length>a1)
2469    912     format(/' ',t8,'bugcheck code',t28,2z2.2)
2470
2471            Call header (lun)
2472            I = 0
2473
2474            If (emb$w_hd_entry .eq. '25'x) then
```

```
2475              Call logger (lun,'FATAL BUGCHECK')
2476              I = 140
2477             Else if (emb$w_hd_entry .eq. '28'x) then
2478              Call logger (lun,'NON-FATAL BUGCHECK')
2479             Else if (emb$w_hd_entry .eq. '70'x) then
2480              call logger (lun,'USER BUGCHECK')
2481             Endif
2482
2483
2484             Call cstring (emb(i+bugchk_code),%REF(bugmsg_text),bugmsg_length)
2485
2486             If (bugmsg_length .ne. 0) then
2487              Call linchk (lun,2)
2488              Write(lun,910) (bugmsg_text(k:k),k = 1,bugmsg_length)
2489             Else
2490              Call linchk (lun,2)
2491              Write(lun,912) (emb(i+bugchk_code+k),k = 1,0,-1)
2492             Endif
2493
2494             Return
2495
2496
2497     C MESSAGE ($SNDERR, OPERATOR, and NETWORK) entries
2498     C
2499     C Output the brief report for the above entry types.
2500     C
2501     39      Continue          ! $snderr message
2502     41      Continue          ! operator message
2503     42      Continue          ! network message
2504
2505             Call header (lun)
2506
2507             If (emb$w_hd_entry .eq. 39) then
2508             Call logger (lun,'$SNDERR MESSAGE')
2509
2510             Else if (emb$w_hd_entry .eq. 41) then
2511             Call logger (lun,'OPERATOR MESSAGE')
2512
2513             Else if (emb$w_hd_entry .eq. 42) then
2514             Call logger (lun,'NETWORK MESSAGE')
2515             Endif
2516
2517             Call linchk (lun,3)
2518
2519             Write(lun,2605)
2520     2605    Format(/' ','MESSAGE TEXT',/)
2521
2522             If (emb$w_ss_msgsz .ne. 0) then
2523
2524             Do 2615,i = 0,(emb$w_ss_msgsz/56)
2525             J = I*56
2526
2527             Call linchk (lun,1)
2528             Write(lun,2610) (emb$t_ss_msgtxt(max(1,j):
2529             1 (j + min(55,emb$w_ss_msgsz - j))))
2530     2610    Format(' ',T8,A)
2531
```

```
2532     2615     Continue
2533
2534              Else
2535
2536              Call linchk (lun,1)
2537              Write(lun,2620)
2538     2620     Format(' ',T8,'[NONE]')
2539              Endif
2540
2541              Return
2542
```

```
2543    C MOUNT/DISMOUNT entries
2544    C
2545    C Output the brief and cryptic reports for the above entry type.
2546    C
2547    64      Continue
2548    65      Continue
2549
2550    C
2551    C Neither mount nor dismount entries are supported for a cryptic
2552    C report so return to the calling routine.
2553    C
2554            If (report_type .EQ. 'C') return
2555
2556    C MOUNT ENTRY
2557    C
2558    C Ensure the volume label is not more then 12 characters in length.
2559    C
2560            If (.NOT. STR$TRIM (emb$t_vm_label,emb$t_vm_label,
2561            1 emb$t_vm_label_length)) then
2562
2563            Emb$t_vm_label_length = 12
2564            Endif
2565
2566    C
2567    C Call the 'mount volume' routine to record this entry in the volume
2568    C label queue.
2569    C
2570            Call HEADER (lun)
2571
2572            If (volume_output) then
2573            If (emb$w_hd_entry .EQ. 64) then
2574
2575            Call LABEL (1,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
2576            1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)
2577
2578    C
2579    C Construct and output the entry header
2580    C
2581            Call LOGGER (lun,'MOUNT VOLUME')
2582            Call LINCHK (lun,4)
2583
2584    C
2585    C Output the entry information.
2586    C
2587            Write(lun,10) emb$t_vm_name,emb$w_vm_unit,
2588            1 emb$t_vm_label(1:emb$t_vm_label_length)
2589    10      format(7' ',t8,'UNIT  ',a<emb$b_vm_namlng>,
2590            1 i<compress4 (lib$extzv(0,16,emb$w_vm_unit))>,':, VOLUME LABEL "',a,
2591            1 '",/)
2592
2593            Write(lun,15) emb$l_vm_oprcnt,emb$l_vm_errcnt
2594    15      format(' ',t8,i<compress4 (emb$l_vm_oprcnt)>,
2595            1 '. QIO OPERATIONS THIS UNIT, ',i<compress4 (emb$l_vm_err:nt)>,
2596            1 '. ERRORS THIS UNIT')
2597
2598
2599    C DISMOUNT ENTRY
```

```
2600     C Determine if this is a dismount entry and a brief report is selected.
2601     C Initialize some counts and call the 'dismount volume' routine to
2602     C search the volume label queue for a matching mount entry. (Mount operation
2603     C and mount error count will be returned if entry found).
2604     C
2605             Else if (emb$w_hd_entry .EQ. 65) then
2606
2607             Call LABEL (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
2608             1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)
2609
2610
2611             Mount_operation_count = -1
2612             Mount_error_count = -1
2613
2614             Call DISMOUNT_VOLUME (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
2615             1 emb$w_vm_unit,emb$t_vm_label,mount_operation_count,
2616             1 mount_error_count)
2617
2618     C
2619     C Construct and output the entry header.
2620     C
2621             Call LOGGER (lun,'DISMOUNT VOLUME')
2622             Call LINCHK (lun,4)
2623
2624     C
2625     C Output the entry information.
2626     C
2627             Write(lun,10) emb$t_vm_name,emb$w_vm_unit,
2628             1 emb$t_vm_label(1:emb$t_vm_label_length)
2629
2630             Write(lun,15) emb$l_vm_oprcnt,emb$l_vm_errcnt
2631
2632     C
2633     C Determine if a matching mount entry was found by checking the mount
2634     C operation and mount error counts. If an entry was found then update
2635     C the volume counts.
2636     C
2637             If (
2638             1 mount_operation_count .ne. -1
2639             1 .and.
2640             1 mount_error_count .ne. -1
2641             1 ) then
2642
2643             Volume_operation_count = emb$l_vm_oprcnt - mount_operation_count
2644             Volume_error_count = emb$l_vm_errcnt - mount_error_count
2645
2646     C
2647     C Via the volume operation and mount counts determine whether to output
2648     C them for the report.
2649     C
2650             If (volume_operation_count) 60,50,50
2651     50      If (volume_error_count) 60,55,55
2652
2653     55      Call LINCHK (lun,1)
2654
2655             Write(lun,56) volume_operation_count,volume_error_count
2656     56      format(' ',t8,i<compress4 (volume_operation_count)>,
```

```
2657            1 '. QIO OPERATIONS THIS VOLUME, ',i<compress4 (volume_error_count)>,
2658            1 '. ERRORS THIS VOLUME')
2659            Endif
2660            Endif
2661
2662            Else            ! volume output = false
2663
2664            if (emb$w_hd_entry .EQ. 64) then
2665
2666            Call MOUNT_VOLUME (1,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
2667            1 emb$w_vm_unit,emb$t_vm_label,emb$l_vm_oprcnt,emb$l_vm_errcnt)
2668
2669            Else if (emb$w_hd_entry .EQ. 65) then
2670
2671            Call DISMOUNT_VOLUME (2,emb$l_hd_sid,emb$b_vm_namlng,emb$t_vm_name,
2672            1 emb$w_vm_unit,emb$t_vm_label,mount_operation_count,
2673            1 mount_error_count)
2674            Endif
2675            Endif
2676
2677    60      Return
2678
```

```
C UNDEFINED INTERRUPT entries
C
C Output the brief and cryptic reports for the above entry type.
C
97      Continue        ! undefined interrupt

        If (report_type .EQ. 'C') return

        Call B_UNDEFINED (lun)
        Return

C LOGSTATUS entry
C
C Output the brief and cryptic reports for the above entry type.
C
99      Continue        ! Log status packet

        If ( Report_type .EQ. 'C' ) Return
        Call header (lun)
        Call logger (lun, 'ERL$LOGSTATUS ENTRY')
        Call dhead2 (lun, 'I/O', emb$b_sp_naming, emb$t_sp_name,
        1 emb$w_sp_unit )
C       Call rollup
        Return


C LOGMESSAGE entry
C
C Output the brief and cryptic reports for the above entry type.
C
100     If ( Report_type .EQ. 'C' ) Return      ! Log message packet
c++
c       Functional description
c
c       This section provides dispatching for entries logged by erl$logmessage.
c       Dispatching is done by emb$w_lm_msgtyp.
c
c--

        Goto (21,22,23,24,25,26,27,28) emb$w_lm_msgtyp

        Tof = 1
        Write1 = 1
        Goto 1010

21      Exit1 = 1                       ! Disk messages
        Goto 1010

22      Exit1 = 1                       ! Tape messages
        Goto 1010


23      Call brief_padriver_logmessage (lun)    ! CI port
        Return

24      Call brief_pudriver_mscp_dispatcher (lun,reccnt) ! UDA port
        Return
```

```
2736
2737     25       tof = 1                          ! Available attention msg
2738              Linchk_P2 = 2
2739              Write1 = 0
2740              Message = msg1 // atten
2741              Length = len(msg1) + len(atten)
2742              Goto 1010
2743
2744     26       Tof = 1                          ! Duplicate unit msg
2745              Linchk_P2 = 2
2746              Write1 = 0
2747              Message = msg2 // atten
2748              Length = len(msg2) + len(atten)
2749              Goto 1010
2750
2751     27       Exit1 = 1                        ! Invalid command log msg
2752              Goto 1010
2753
2754     28       Tof = 1                          ! Access path attention msg
2755              Linchk_P2 = 2
2756              Write1 = 0
2757              Message = msg3 // atten
2758              Length = len(msg3) + len(atten)
2759
2760
2761     1010     If ( tof .eq. 1 ) then
2762                Call frctof (lun)
2763                Tof = 0
2764              Endif
2765
2766              Call header (lun)
2767              Call logger (lun,'ERL$LOGMESSAGE ENTRY')
2768              Call dhead2 (lun,'I/O',emb$b_lm_namlng,emb$t_lm_name,emb$w_lm_unit)
2769              If ( exit1 .eq. 1 ) then
2770                Exit1 = 0
2771                Return
2772              Endif
2773
2774              If ( linchk_P2 .ne. 0 ) then
2775                Call linchk ( lun, linchk_p2 )
2776                Linchk_P2 = 0
2777              Endif
2778
2779
2780              If ( Write1 .EQ. 1 ) then
2781                Write(lun,2590) 'EMB$W_LM_MSGTYP',(emb$w_lm_msgtyp,i = 1,2)
2782     2590       format(/' ',t8,a,t28,z4.4,/,
2783              1 t40,a,i<compress4 (lib$extzv(0,16,emb$w_lm_msgtyp))>)
2784                Call dumpreg (lun,((record_length + 3)/4) - 9,emb(38))
2785
2786              Else
2787
2788                Write(lun,2510) Message
2789     2510       Format(/' ',a<length>)
2790              Endif
2791
2792              Return
```

```
2793
2794
2795
2796    C LOGMSCP (mscp message without ucb)
2797    C
2798    C Output the brief report for the above entry type.
2799    C
2800    101     If (report_type .EQ. 'C') return
2801
2802            Call HEADER (lun)
2803            Call LOGGER (lun,'ERL$LOGMSCP ENTRY')
2804
2805            if (logmscp_type .eq. 9) then
2806
2807            Call LINCHK (lun,2)
2808            write(lun,30) 'UNEXPECTED END MESSAGE ''MSCP$W_STATUS'''
2809    30      format(/' ',t8,a)
2810
2811            else if (logmscp_type .eq. 10) then
2812
2813            Call LINCHK (lun,2)
2814            write(lun,30) 'UNEXPECTED ATTENTION CONTROL MESSAGE'
2815
2816            else if (logmscp_type .eq. 11) then
2817
2818            Call LINCHK (lun,2)
2819            write(lun,30) 'DATAGRAM FOR NON-EXISTING ''UCB'''
2820            endif
2821
2822            return
2823
2824
2825
2826    C MEMORY entries
2827    C
2828    C Output the brief and cryptic reports for the above entry type.
2829    C Output the entry header.
2830    C
2831    5       Continue
2832    6       Continue
2833    8       Continue
2834
2835    C
2836    C Determine if a summary report should be output. If so,
2837    C execute the necessary code to produce a memory summary report.
2838    C
2839
2840            If ((memory_summ) .OR. (all_summ)) then
2841              Call EXEC_IMAGE(xfer_addr,lun,'R')
2842            Endif
2843
2844            If (report_type .NE. 'C') then
2845            Call HEADER (lun)
2846    C SBI Alert
2847            If (emb$w_hd_entry .EQ. 5) then
2848            Call LOGGER (lun,'SBI ALERT')
2849
```

```
2850    C Hard ECC
2851            Else if (emb$w_hd_entry .EQ. 8) then
2852            Call LOGGER (lun,'FATAL MEMORY ERROR')
2853
2854    C Soft ECC
2855            Else
2856            Call LOGGER (lun,'MEMORY ERROR')
2857            Endif
2858
2859            Endif
2860
2861    C Determine the cpu type.
2862    C
2863            J = 0
2864            Cpu_type = LIB$EXTZV(24,8,emb$l_hd_sid)
2865
2866            If ( cpu_type .EQ. 255
2867            1 .OR.
2868            2 cpu_type .EQ. 1 ) then
2869
2870    C 11/780 cpu
2871    C  Interpret the information for all controllers in this entry.
2872    C  Determine the memory type.
2873    C
2874            Do 635, I=1, controller_count
2875            Memory_type = LIB$EXTZV(5,3,buffer(2+J))
2876
2877    C MS780C device dependent routine
2878    C
2879    610     If (memory_type .EQ. 0) then
2880
2881    C Copy the memory registers into mem_reg_buffer
2882    C
2883            Call MOVC3 (%val(12),buffer(2+j),mem_reg_buffer)
2884
2885    C Determine if it should be a brief report. Pass the memory registers,
2886    C register herald and the adapter tr to the brief routine for output.
2887    C
2888            If (report_type .EQ. 'B') then
2889            Call BRIEF32 (lun,(3),memory_register_a,ms780c_reg_herald,'TR ',
2890            1 buffer(1+j))
2891
2892            Else
2893
2894    C Determine if it should be a  register dump report. Pass the memory
2895    C registers, register herald and the adapter tr to the brief routine
2896    C for output.
2897    C
2898            Call CRYPTK (lun,32,(3),memory_register_a,ms780c_reg_herald,'TR ',
2899            1 buffer(1+j))
2900            Endif
2901
2902            J=J+4
2903            Endif
2904
2905
2906    C  MA780 device dependent routine
```

```
2907    C
2908    620     If (memory_type .EQ. 2) then
2909
2910    C Copy the memory registers into mem_reg_buffer
2911    C
2912            Call MOVC3 (%val(32),buffer(2+j),mem_reg_buffer)
2913
2914    C Determine if it should be a brief report. Pass the memory registers,
2915    C register herald and the adapter tr to the brief routine for output.
2916    C
2917            If (report_type .EQ. 'B') then
2918            Call BRIEF32 (lun,(8),port_configuration_register,ma780_reg_herald,
2919            1 'TR ',buffer(1+j))
2920
2921            Else
2922
2923    C Determine if it should be a  register dump report. Pass the memory
2924    C registers, register herald and the adapter tr to the brief routine
2925    C for output.
2926    C
2927            Call CRYPTK (lun,32,(8),port_configuration_register,ma780_reg_herald,
2928            1 'TR ',buffer(1+j))
2929            Endif
2930
2931            J=J+9
2932            Endif
2933
2934
2935    C MS780E device dependent routine
2936    C
2937    630     If (memory_type .EQ. 3) then
2938
2939    C Copy the memory registers into mem_reg_buffer
2940    C
2941            Call MOVC3 (%val(16),buffer(2+j),mem_reg_buffer)
2942
2943    C Determine if it should be a brief report. Pass the memory registers,
2944    C register herald and the adapter tr to the brief routine for output.
2945    C
2946            If (report_type .EQ. 'B') then
2947            Call BRIEF32 (lun,(4),memory_register_a,ms780e_reg_herald,'TR ',
2948            1 buffer(1+j))
2949
2950            Else
2951
2952    C Determine if it should be a  register dump report. Pass the memory
2953    C registers, register herald and the adapter tr to the brief routine
2954    C for output.
2955    C
2956            Call CRYPTK (lun,32,(4),memory_register_a,ms780e_reg_herald,'TR ',
2957            1 buffer(1+j))
2958            Endif
2959
2960            J=J+5
2961            Endif
2962
2963    635     CONTINUE
```

```
2964            Return
2965
2966            Else if (cpu_type .EQ. 2 ) then
2967
2968    C 11/750 cpu
2969    C   MS750 device dependent routine
2970    C
2971    640     Call MOVC3 (%val(12),memory_registers,mem_reg_buffer)
2972
2973            If (report_type .EQ. 'B') then
2974
2975            Call BRIEF32 (lun,(3),memory_register_0,mem_reg_herald,'SI ',
2976            1 slot_index)
2977
2978            Else
2979
2980            Call CRYPTK (lun,32,(3),memory_register_0,mem_reg_herald,'SI ',
2981            1 slot_index)
2982
2983            Endif
2984            Return
2985
2986
2987            Else if ( cpu_type .EQ. 3 ) then
2988
2989    C 11/730 cpu
2990    C  MS730 device dependent routine
2991    C
2992    650     Call MOVC3 (%val(12),memory_register_0,mem_reg_buffer)
2993            If (report_type .EQ. 'B') then
2994
2995            Call BRIEF32 (lun,(3),memory_csr0,mem_reg_herald,'MEM',0)
2996
2997            Else
2998
2999            Call CRYPTK (lun,32,(3),memory_csr0,mem_reg_herald,'MEM',0)
3000            Endif
3001
3002            Endif
3003
3004            Return
3005
```

```
3006     C
3007     C Determine the report type and call the brief and/or cryptic routines.
3008     C Parameters for call to brief are:
3009     C (lun,number of registers,registers,register heralds,name,unit)
3010     C
3011     1000    If ( Report_type .EQ. 'B' ) then
3012                Call DHEAD1 (lun, Dhead_P2 )
3013                If ( Brief_32 .EQ. 1 ) then
3014                     Call BRIEF32 (lun, Brief_P2, %val(Brief_P3), Brief_P4,
3015          1                        emb$t_dv_name, emb$w_dv_unit)
3016                     Brief_32 = 0
3017                Else
3018                     Call BRIEF16 (lun, Brief_P2, %val(Brief_P3), Brief_P4,
3019          1                        emb$t_dv_name, emb$w_dv_unit)
3020                Endif
3021             Else
3022                Call CRYPTK (lun, Cryp_P2, Brief_P2, %val(Brief_P3), Brief_P4,
3023          1                  emb$t_dv_name, emb$w_dv_unit)
3024             Endif
3025             Return
3026
3027     999     Return
3028
3029             End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	6084	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	1571	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	4020	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
4	SYECOM	44	PIC OVR REL GBL SHR NOEXE RD WRT LONG
5	MODE	55	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	12286	

ENTRY POINTS

Address	Type	Name
0-00000000		BRIEF_C_DISPATCHER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000028	I*4	ADAPTER_TR	2-000005B8	I*4	ALL_SUMM
2-000004E1	L*1	BRIEF_32	2-000004DF	L*1	BRIEF_P2
2-0000059C	I*4	BRIEF_P3	2-000005C8	I*4	BUGMSG_LENGTH
2-00000550	CHAR	BUGMSG_TEXT	2-0000009F	I*4	CMD$A_POINTER
2-0000009E	L*1	CMD$B_CLASS	2-0000009D	L*1	CMD$B_TYPE
2-0000009B	I*2	CMD$W_SIZE	3-00000052	I*2	CODE_WORD
3-00000010	I*4	CONTROLLER_COUNT	2-000005A4	I*4	CPU_TYPE
4-00000012	L*1	CP_11750	4-00000011	L*1	CP_T1780
4-00000013	L*1	CP_11ZZZ	4-00000014	L*4	CRYPTK_FLAG
2-000004E0	L*1	CRYP_P2	2-00000024	I*4	DEV$A_BLINK
2-00000020	I*4	DEV$A_FLINK	2-0000003D	L*1	DEV$B_CLASS
2-00000028	L*1	DEV$B_DEV_NAME_LENGTH	2-00000029	CHAR	DEV$T_DEV_NAME
2-00000029	I*2	DEV$T_DEV_NAME_LEN	2-0000002B	CHAR	DEV$T_DEV_NAME_TXT
2-0000003B	I*2	DEV$W_UNIT	2-000005E4	R*4	DEVICE_CLASS
2-000005E8	R*4	DEVICE_TYPE	2-000004E5	L*1	DEVICE_TYPEDEVICE_CLASS
4-0000000D	I*4	DEV_CHAR	3-0000001C	L*1	EMB$B_DV_CLASS
3-00000010	L*1	EMB$B_DV_ERTCNT	3-00000011	L*1	EMB$B_DV_ERTMAX
3-0000003E	L*1	EMB$B_DV_NAMLNG	3-0000003A	L*1	EMB$B_DV_SLAVE
3-0000001D	L*1	EMB$B_DV_TYPE	3-00000010	L*1	EMB$B_LM_CLASS
3-00000014	L*1	EMB$B_LM_NAMLNG	3-00000011	L*1	EMB$B_LM_TYPE
3-00000010	L*1	EMB$B_SP_CLASS	3-00000040	L*1	EMB$B_SP_NAMLNG
3-00000011	L*1	EMB$B_SP_TYPE	3-0000001E	L*1	EMB$B_VM_NAMLNG
3-00000036	I*4	EMB$L_DV_CHAR	3-00000012	I*4	EMB$L_DV_IOSB1
3-00000016	I*4	EMB$L_DV_IOSB2	3-00000026	I*4	EMB$L_DV_MEDIA
3-0000004E	I*4	EMB$L_DV_NUMREG	3-0000002E	I*4	EMB$L_DV_OPCNT
3-00000032	I*4	EMB$L_DV_OWNUIC	3-0000001E	I*4	EMB$L_DV_RQPID
3-00000000	I*4	EMB$L_HD_SID	3-00000014	I*4	EMB$L_SP_BCNT
3-00000038	I*4	EMB$L_SP_CHAR	3-0000003C	I*4	EMB$L_SP_CMDREF
3-00000020	I*4	EMB$L_SP_IOSB1	3-00000024	I*4	EMB$L_SP_IOSB2
3-00000018	I*4	EMB$L_SP_MEDIA	3-0000002C	I*4	EMB$L_SP_OPCNT
3-00000034	I*4	EMB$L_SP_OWNUIC	3-0000001C	I*4	EMB$L_SP_RQPID
3-00000014	I*4	EMB$L_VM_ERRCNT	3-00000018	I*4	EMB$L_VM_OPRCNT

```
  3-00000010  I*4  EMB$L_VM_OWNUIC
  3-00000015  CHAR EMB$T_LM_NAME
  3-00000012  CHAR EMB$T_SS_MSGTXT
  2-000005D0  I*4  EMB$T_VM_LABEL_LENGTH
  3-00000024  I*2  EMB$W_DV_BCNT
  3-0000002C  I*2  EMB$W_DV_ERRCNT
  3-0000001A  I*2  EMB$W_DV_STS
  3-00000004  I*2  EMB$W_HD_ENTRY
  3-00000024  I*2  EMB$W_LM_MSGTYP
  3-00000012  I*2  EMB$W_SP_BOFF
  3-00000028  I*2  EMB$W_SP_FUNC
  3-0000002A  I*2  EMB$W_SP_UNIT
  3-00000030  I*2  EMB$W_VM_NUMSET
  3-0000002E  I*2  EMB$W_VM_VOLNUM
  4-0000001D  L*1  EOF_FLAG
  2-00000063  I*4  ERL$L_END_ENTRY
  2-0000006B  I*4  ERL$L_MBX_UNIT
  2-0000005F  I*4  ERL$L_START_ENTRY
  2-000004E3  L*1  EXIT1
  2-000005BC  I*4  I
  2-000005C0  I*4  J
  2-000005A8  I*4  LENGTH
  4-0000000C  L*1  LINES
  4-00000027  I*4  LSTLUN
  4-0000001F  I*4  MAILBOX_CHANNEL
  3-00000018  I*4  MEMORY_REGISTERS
  2-00000000  I*4  MEMORY_REGISTER_A
  2-000005F8  I*4  MEMORY_TYPE
  2-000005D8  I*4  MOUNT_ERROR_COUNT
  4-0000002B  CHAR OPTIONS
  3-00000056  I*4  PCNFGR
  4-00000008  L*4  PRINTER
 AP-0000000C@ I*4  RECORD_LENGTH
  2-000005F4  R*4  REG_HERALD
  2-000005CC  I*4  SLOT_INDEX
 AP-00000014@ I*4  SUMMARY_FLAG
  2-000005EC  R*4  THENJ
  2-0000054D  CHAR TR
  3-00000010  I*4  UBA_CSR
  3-00000020  I*4  UBA_FMER
  3-00000018  I*4  UBA_SR
  3-0000005A  I*4  UCB$L_BS_CUR
  4-00000019  L*1  VALID_CLASS
  4-0000001B  L*1  VALID_ENTRY
  2-000005E0  I*4  VOLUME_ERROR_COUNT
  4-00000018  L*1  VOLUME_OUTPUT
  2-000005AC  I*4  XFER_ADDR
  3-0000003F  CHAR EMB$T_DV_NAME
  3-00000041  CHAR EMB$T_SP_NAME
  3-00000032  CHAR EMB$T_VM_LABEL
  3-0000001F  CHAR EMB$T_VM_NAME
  3-00000022  I*2  EMB$W_DV_BOFF
  3-0000003C  I*2  EMB$W_DV_FUNC
  3-0000002A  I*2  EMB$W_DV_UNIT
  3-0000000E  I*2  EMB$W_HD_ERRSEQ
  3-00000012  I*2  EMB$W_LM_UNIT
  3-00000030  I*2  EMB$W_SP_ERRCNT
  3-00000032  I*2  EMB$W_SP_STS
  3-00000010  I*2  EMB$W_SS_MSGSZ
  3-0000001C  I*2  EMB$W_VM_UNIT
  4-0000001E  L*1  ENV_VALUE
  2-0000004A  L*1  ERL$B_RPT_TYPE
  2-00000067  I*4  ERL$L_MBX_CHANNEL
  2-0000004B  I*4  ERL$L_SID_SELECTION
  2-0000006F  CHAR ERL$T_GETCHN_BUFFER
  4-00000004  L*4  FORMS
  2-00000536  CHAR IMAGE_NAME
  2-000005C4  I*4  K
  2-000005A0  I*4  LINCHK_P2
  3-00000010  I*2  LOGMSCP_TYPE
 AP-00000004@ L*1  LUN
  2-00000000  I*4  MEMORY_CSR0
  2-00000000  I*4  MEMORY_REGISTER_0
  2-000005B4  I*4  MEMORY_SUMM
  2-000004E6  CHAR MESSAGE
  2-000005D4  I*4  MOUNT_OPERATION_COUNT
  3-00000052  I*4  PADRIVER_ERROR_TYPE_CODE
  2-00000000  I*4  PORT_CONFIGURATION_REGISTER
  4-00000000  I*4  RECCNT
  4-00000023  I*4  RECORD_SIZE
 AP-00000008@ CHAR REPORT_TYPE
  2-000005B0  I*4  STATUS
 AP-00000010@ L*1  SUMMARY_QUAL
  2-000004E2  L*1  TOF
  3-00000014  I*4  UBA_CR
  3-0000001C  I*4  UBA_DCR
  3-00000024  I*4  UBA_FUBAR
  3-00000052  I*4  UCB$B_BS_ERRMSG
  2-000005F0  R*4  UDA_SA
  4-0000001A  L*1  VALID_CPU
  4-0000001C  L*1  VALID_TYPE
  2-000005DC  I*4  VOLUME_OPERATION_COUNT
  2-000004E4  L*1  WRITE1
```

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-000000A4	I*4	BRIEF_P4	8	(2)
3-00000010	I*4	BUFFER	484	(0:120)
2-00000049	L*1	BUG$R_BUGCHK_FLAGS	1	(1)
2-00000049	L*1	BUGCHK_DEF	1	(1)

2-0000040D	CHAR	CI_REG_HERALD	36	(0:5)
2-000001A2	CHAR	DD_REG_HERALD	20	(0:3)
2-0000003E	L*1	DEV$R_FLAGS_OVERLAY	1	(1)
2-0000003E	L*1	DEV$R_FLAG_BITS	1	(1)
2-00000020	L*1	DEV_QUEUE	31	(31)
2-000000AC	I*4	DHEAD_P2	8	(2)
2-000003A1	CHAR	DR11W_REG_HERALD	20	(0:4)
2-00000371	CHAR	DR750_REG_HERALD	48	(0:5)
2-00000359	CHAR	DR780_REG_HERALD	24	(0:5)
2-00000403	CHAR	DT07_REG_HERALD	10	(0:1)
2-000002E2	CHAR	DUP_REG_HERALD	25	(0:4)
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB$B_LM_MSGTXT	460	(460)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
2-0000004F	I*4	ERL$Q_END_DATE	8	(2)
2-00000057	I*4	ERL$Q_START_DATE	8	(2)
2-0000009B	L*1	ERL$R_CMD_LINE	8	(8)
2-00000043	L*1	EXC$R_EXCLUDE_FLAGS	3	(3)
2-00000043	L*1	EXCDEF	3	(3)
3-0000006E	I*4	EXTENDED_REGISTERS	8	(2)
3-0000006E	I*2	EXTENDED_STATUS	8	(4)
2-00000040	L*1	INC$R_INCLUDE_FLAGS	3	(3)
2-00000040	L*1	INCDEF	3	(3)
3-0000005E	I*2	INITIALIZATION_HANDSHAKE	16	(8)
2-00000317	CHAR	KMS3271_REG_HERALD	50	(0:9)
2-00000349	CHAR	LPA_REG_HERALD	16	(0:3)
2-0000047C	CHAR	MA780_REG_HERALD	64	(8)
2-00000455	CHAR	MBA_REG_HERALD	24	(6)
2-00000000	I*4	MEM_REG_BUFFER	32	(8)
2-0000046D	CHAR	MEM_REG_HERALD	15	(3)
2-0000027C	CHAR	MF_REG_HERALD	102	(0:16)
2-000001B6	CHAR	ML_REG_HERALD	60	(0:11)
5-00000000	CHAR	MODES	55	(0:4)
2-000004BC	CHAR	MS780C_REG_HERALD	15	(3)
2-000004CB	CHAR	MS780E_REG_HERALD	20	(4)
3-0000008E	I*4	MSG_BUF	28	(0:6)
2-00000046	L*1	OPT$R_OPT_FLAGS	3	(3)
2-00000046	L*1	OPTDEF	3	(3)
2-000003B5	CHAR	PCL11R_REG_HERALD	36	(0:5)
2-000003D9	CHAR	PCL11T_REG_HERALD	42	(0:6)
2-0000004A	L*1	PRS_DATA	81	(81)
2-0000009B	L*1	PRS_TABLE	8	(8)
2-000001F2	CHAR	RB_REG_HERALD	56	(0:6)
2-000000B4	I*4	REG_COPY	48	(0:11)
2-00000114	CHAR	REG_HERALD1	48	(0:11)
2-00000144	CHAR	REG_HERALD2	48	(0:11)
3-00000010	I*4	RH780_CONTROL_REGISTER	24	(6)
2-000000E4	CHAR	RK_REG_HERALD	48	(0:11)
2-00000174	CHAR	RL_REG_HERALD	16	(0:3)
2-00000184	CHAR	RX_REG_HERALD	30	(0:5)
2-000002FB	CHAR	SB_REG_HERALD	28	(0:3)
2-0000003F	L*1	SUM$R_SUMMARY_FLAGS	1	(1)
2-0000003F	L*1	SUMDEF	1	(1)
2-0000022A	CHAR	TAPE_REG_HERALD	40	(0:9)
2-00000252	CHAR	TS_REG_HERALD	42	(0:6)
2-00000431	CHAR	UBA_REG_HERALD	36	(0:5)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
0-00000111	1	0-00000A31	2	0-00000A67	4	0-000013D3	5	0-000013D3	6	0-00000A67	7
0-000013D3	8	0-00000AEA	9	1-00000441	10'	0-00000B44	11	0-00000B9E	12	1-00000474	15'
0-00001167	21	0-0000116F	22	0-00001177	23	0-00001187	24	0-00001197	25	0-000011B9	26
0-000011DB	27	0-000011E2	28	1-00000522	30'	0-00000BF8	32	**	34	0-00000C3A	35
0-00000BF8	36	0-00000C7C	37	0-00000C3A	38	0-00000D9A	39	0-00000C7C	40	0-00000D9A	41
0-00000D9A	42	**	50	**	55	1-000004B8	56'	0-000010EB	60	0-00000ECD	64
0-00000ECD	65	0-00000111	96	0-000010EC	97	0-00000111	98	0-00001104	99	0-0000113A	100
0-000012F7	101	0-00000C7C	112	0-000003FB	201	0-000003FB	202	0-000003FB	203	0-00000429	204
0-0000048B	205	0-000004E6	303	0-00000517	307	0-0000057B	318	0-000005C2	401	0-000005F0	402
0-0000064D	403	0-000006A9	404	0-000006D7	405	0-00000705	406	**	407	0-00000733	408
0-000007B1	501	0-000007B1	502	0-0000081A	503	0-00000824	504	**	610	**	620
**	630	**	635	**	640	**	650	1-000003EF	910'	1-000003FB	912'
0-000016DB	999	0-00001676	1000	0-00001202	1010	1-00000517	2510'	1-00000500	2590'	1-00000418	2605'
1-0000042C	2610'	**	2615	1-00000433	2620'	0-000008C6	5001	0-000008DB	5002	0-000008F0	5003
0-00000904	5004	0-00000918	5005	0-0000092A	5006	0-0000097A	5007	1-00000375	5210'	1-000003B8	5215'
1-000003D8	5220'	0-00000152	10000	0-00000181	10001	0-00000181	10002	0-000001AF	10003	0-000001AF	10004
0-000001AF	10005	0-000001DD	10006	0-000001AF	10007	0-0000020B	10009	0-0000020B	10010	0-00000239	10011
0-00000239	10012	0-000001DD	10013	0-000002AF	10014	0-000001DD	10015	0-00000332	10017	0-000003B5	10018
0-000003B5	10019	0-000003E8	20000	0-000004B9	30000	0-000005A9	40000	0-0000079C	50000		

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
	BRIEF16		BRIEF32		BRIEF_PADRIVER_LOGMESSAGE
	BRIEF_PUDRIVER_MSCP_DISPATCHER		B_UNDEFINED	I*4	COMPRESS4
	CRYPTR		CSTRING		DHEAD1
	DHEAD2		DISMOUNT_VOLUME		DUMPREG
	ERF_LOADERR		EXEC_IMAGE		FRCTOF
	HEADER	I*4	IMAGE_LOADER		LABEL
I*4	LIB$EXTZV	I*4	LIB$LOCC		LIB$SIGNAL
	LINCHK		LOGGER		MOUNT_VOLUME
	MOVC3		PADRIVER_ATTENTION_ERROR_CODE	L*1	STR$TRIM
	UCB$$B_BS_ERRMSG				

```
0001            Subroutine BRIEF_PADRIVER_LOGMESSAGE (lun)
0002
0003
0004            include 'src$:msghdr.for /nolist'
0063            include 'src$:emblmdef.for /nolist'
0132
0133            byte            lun
0134
0135            equivalence     (emb$b_lm_msgtxt(1),padriver_error_type_code)
0136
0137            integer*4       padriver_error_type_code
0138            integer*4       error_subtype
0139            integer*4       compress4
0140            integer*4       error_type
0141
0142
0143            call header (lun)
0144            call logger (lun,'ERL$LOGMESSAGE ENTRY')
0145
0146            error_subtype = lib$extzv(0,8,padriver_error_type_code)
0147            error_type = lib$extzv(8,7,padriver_error_type_code)
0148
0149            call linchk (lun,2)
0150
0151            if (error_type .eq. 64) then
0152
0153               if (error_subtype .eq. 0) then
0154
0155                 write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0156           1      emb$w_lm_unit,'UNRECOGNIZED ''SCA'' PACKET'
0157        10       format(/' ','CI SUB-SYSTEM, ',a<emb$b_lm_namlng>,
0158           1      i<compress4 (lib$extzv(0,16,emb$w_lm_unit))>,': - ',
0159           1      a,:i<compress4 (error_subtype)>,:a)
0160
0161               else if (error_subtype .eq. 1) then
0162
0163                      write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0164           1          emb$w_lm_unit,'PORT HAS CLOSED ''VIRTUAL CIRCUIT'''
0165
0166               else if (error_subtype .eq. 2) then
0167
0168                      write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0169           1          emb$w_lm_unit,'SOFTWARE SHUTTING DOWN PORT'
0170
0171               else if (error_subtype .eq. 3) then
0172
0173                      write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0174           1          emb$w_lm_unit,'SOFTWARE IS CLOSING ''VIRTUAL CIRCUIT'''
0175
0176               else if (error_subtype .eq. 4) then
0177
0178                      write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0179           1          emb$w_lm_unit,'RECEIVED ''CONNECT'' WITHOUT PATH-BLOCK'
0180
0181               else if (error_subtype .eq. 5) then
0182
0183                      write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
```

```
0184            1          emb$w_lm_unit,'INAPPROPRIATE "SCA" CONTROL MESSAGE'
0185
0186               else if (error_subtype .eq. 6) then
0187
0188                     write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0189            1         emb$w_lm_unit,'NO PATH-BLOCK DURING "VIRTUAL CIRCUIT" CLOSE'
0190
0191                  else if (error_subtype .eq. 7) then
0192                     write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0193            1        emb$w_lm_unit,'HSC ERROR LOGGING DATAGRAM RECEIVED.'
0194
0195                  else if (error_subtype .eq. 8) then
0196                     write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0197            1        emb$w_lm_unit,'REMOTE SYSTEM CONFLICTS WITH KNOWN SYSTEM.'
0198
0199
0200               endif
0201
0202            else If (error_type .eq. 65) then
0203               Write(lun,10) emb$t_lm_name(1:emb$b_lm_namlng),
0204            1  emb$w_lm_unit,'DATA CABLE(S) CHANGE OF STATE'
0205               Call Tinchk (lun,2)
0206               If (error_subtype .eq. 0) then
0207                 write(lun,12) 'PATH #0. HAS GONE FROM GOOD TO BAD'
0208    12           format(/' ',t8,a)
0209               Else
0210                 Goto (200,205,210,215,220,225,230,235,240,245,250)
0211            1    error_subtype
0212    999          Continue
0213              Endif
0214            else
0215
0216            write(lun,25) emb$t_lm_name(1:emb$b_lm_namlng),
0217            1 emb$w_lm_unit,'"PADRIVER" ERROR TYPE #',error_type,
0218            1 '., ERROR SUB-TYPE #',error_subtype,'.'
0219    25      format(/' ','CI SUB-SYSTEM, ',a<emb$b_lm_namlng>,
0220            1 i<compress4 (lib$extzv(0,16,emb$w_lm_unit))>,': - ',
0221            1 a,i<compress4 (error_type)>,a,i<compress4 (error_subtype)>,a)
0222            endif
0223
0224            call padriver_initialization (lun,padriver_error_type_code)
0225
0226            Return
0227    200          write(lun,12) 'PATH #1. HAS GONE FROM GOOD TO BAD'
0228                 goto 999
0229
0230    205          write(lun,12) 'PATH #0. HAS GONE FROM BAD TO GOOD'
0231                 goto 999
0232
0233    210          write(lun,12) 'PATH #1. HAS GONE FROM BAD TO GOOD'
0234                 goto 999
0235
0236    215          write(lun,12) 'CABLES HAVE GONE FROM UNCROSSED TO CROSSED'
0237                 goto 999
0238
0239    220          write(lun,12) 'CABLES HAVE GONE FROM CROSSED TO UNCROSSED'
0240                 goto 999
```

```
0241
0242     225          write(lun,12) 'PATH #0. LOOPBACK HAS GONE FROM GOOD TO BAD'
0243                  goto 999
0244
0245     230          write(lun,12) 'PATH #1. LOOPBACK HAS GONE FROM GOOD TO BAD'
0246                  goto 999
0247
0248     235          write(lun,12)'PATH #0. LOOPBACK HAS BECOME GOOD, UNCROSSED'
0249                  goto 999
0250
0251     240          write(lun,12)'PATH #1. LOOPBACK HAS BECOME GOOD, UNCROSSED'
0252                  goto 999
0253
0254     245          write(lun,12)
0255              1    'PATH #0. HAS BECOME WORKING BUT CROSSED TO PATH #1.'
0256                  goto 999
0257
0258     250          write(lun,12)
0259              1    'PATH #1. HAS BECOME WORKING BUT CROSSED TO PATH #0.'
0260                  goto 999
0261             End


PROGRAM SECTIONS

    Name                                Bytes   Attributes

  0 $CODE                               1625   PIC CON REL LCL   SHR   EXE   RD NOWRT LONG
  1 $PDATA                              1040   PIC CON REL LCL   SHR NOEXE   RD NOWRT LONG
  2 $LOCAL                               332   PIC CON REL LCL NOSHR NOEXE   RD   WRT LONG
  3 EMB                                  512   PIC OVR REL GBL   SHR NOEXE   RD   WRT LONG

    Total Space Allocated               3509

ENTRY POINTS

    Address  Type  Name

  0-00000000       BRIEF_PADRIVER_LOGMESSAGE

VARIABLES

    Address  Type  Name                            Address  Type  Name

  3-00000010  L*1  EMB$B_LM_CLASS                 3-00000014  L*1  EMB$B_LM_NAMLNG
  3-00000011  L*1  EMB$B_LM_TYPE                  3-00000000  I*4  EMB$L_HD_SID
  3-00000015  CHAR EMB$T_LM_NAME                  3-00000004  I*2  EMB$W_HD_ENTRY
  3-0000000E  I*2  EMB$W_HD_ERRSEQ                3-00000024  I*2  EMB$W_LM_MSGTYP
  3-00000012  I*2  EMB$W_LM_UNIT                  2-00000000  I*4  ERROR_SUBTYPE
  2-00000004  I*4  ERROR_TYPE                    AP-00000004@ L*1  LUN
  3-00000026  I*4  PADRIVER_ERROR_TYPE_CODE
```

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB$B_LM_MSGTXT	460	(460)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-0000039D	10'	1-000003D0	12'	1-000003D8	25'	0-00000435	200	0-0000045B	205	0-00000481	210
0-000004A7	215	0-000004CD	220	0-000004F3	225	0-00000519	230	0-0000053F	235	0-00000565	240
0-0000058B	245	0-000005B1	250	0-000003AD	999						

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name
I*4	COMPRESS4		HEADER	I*4	LIB$EXTZV
	LINCHK		LOGGER		PADRIVER_INITIALIZATION

CL

01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01
01

PR

EN

VA

```
0001
```

```
0002            Subroutine BRIEF_PUDRIVER_MSCP_DISPATCHER (lun,reccnt)
0003
0004            include 'src$:msghdr.for /nolist'
0063            include 'src$:emblmdef.for /nolist'
0132            include 'src$:embspdef.for /nolist'
0245
0246            byte            lun
0247            byte            mslg$b_format
0248
0249            Integer*4       reccnt
0250
0251            equivalence     (emb(46),mslg$b_format)
0252
0253
0254            if (emb$w_hd_entry .eq. 100) then
0255
0256               call FRCTOF (lun)
0257               call HEADER2 (lun,reccnt)
0258               call LOGGER (lun,'ERL$LOGMESSAGE ENTRY')
0259               call DHEAD2 (lun,'"DSA" PORT',emb$b_lm_namlng,emb$t_lm_name,
0260            1               emb$w_lm_unit)
0261
0262            else if (emb$w_hd_entry .eq. 99) then
0263
0264                    call FRCTOF (lun)
0265                    call HEADER2 (lun,reccnt)
0266                    call LOGGER (lun,'ERL$LOGSTATUS ENTRY')
0267                    call DHEAD2 (lun,'"DSA" PORT',emb$b_sp_namlng,emb$t_sp_name,
0268            1                    emb$w_sp_unit)
0269
0270            Endif
0271
0272            Return
0273            End
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	148	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	52	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	132	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	844	

ENTRY POINTS

Address	Type	Name
0-00000000		BRIEF_PUDRIVER_MSCP_DISPATCHER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000010	L*1	EMB$B_LM_CLASS	3-00000014	L*1	EMB$B_LM_NAMLNG
3-00000011	L*1	EMB$B_LM_TYPE	3-00000010	L*1	EMB$B_SP_CLASS
3-00000040	L*1	EMB$B_SP_NAMLNG	3-00000011	L*1	EMB$B_SP_TYPE
3-00000000	I*4	EMB$L_HD_SID	3-00000014	I*4	EMB$L_SP_BCNT
3-00000038	I*4	EMB$L_SP_CHAR	3-0000003C	I*4	EMB$L_SP_CMDREF
3-00000020	I*4	EMB$L_SP_IOSB1	3-00000024	I*4	EMB$L_SP_IOSB2
3-00000018	I*4	EMB$L_SP_MEDIA	3-0000002C	I*4	EMB$L_SP_OPCNT
3-00000034	I*4	EMB$L_SP_OWNUIC	3-0000001C	I*4	EMB$L_SP_RQPID
3-00000015	CHAR	EMB$T_LM_NAME	3-00000041	CHAR	EMB$T_SP_NAME
3-00000004	I*2	EMB$W_HD_ENTRY	3-0000000E	I*2	EMB$W_HD_ERRSEQ
3-00000024	I*2	EMB$W_LM_MSGTYP	3-00000012	I*2	EMB$W_LM_UNIT
3-00000012	I*2	EMB$W_SP_BOFF	3-00000030	I*2	EMB$W_SP_ERRCNT
3-00000028	I*2	EMB$W_SP_FUNC	3-00000032	I*2	EMB$W_SP_STS
3-0000002A	I*2	EMB$W_SP_UNIT	AP-0000004@	L*1	LUN
3-0000002E	L*1	MSLG$B_FORMAT	AP-0000008@	I*4	RECCNT

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000026	L*1	EMB$B_LM_MSGTXT	460	(460)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name
	DHEAD2		FRCTOF		HEADER2		LOGGER

0001

COMMAND QUALIFIERS

FORTRAN /LIS=LIS$:B_C_DISP/OBJ=OBJ$:B_C_DISP MSRC$:B_C_DISP

/CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
/DEBUG=(NOSYMBOLS,TRACEBACK)
/STANDARD=(NOSYNTAX,NOSOURCE_FORM)
/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE /NOMACHINE_CODE /CONTINUATIONS=19

COMPILATION STATISTICS

Run Time: 30.87 seconds
Elapsed Time: 81.18 seconds
Page Faults: 595
Dynamic Memory: 455 pages

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

ERFPROC4
MAP

EMBETDEF
REQ

BUGCHK
LIS

DEVERR
FOR

MODES
FOR

SSMESSAGE
FOR

B_C_DISP
LIS

ERFSHR
MAP

BSDRIVER
LIS

ERFPROC3
MAP

ERFDEF
REQ

MSGHDR
FOR

SYECOM
FOR

ERFRLTIM
MAP

ERFTAPE
MAP

PARSERDAT
SDL

DEVICLASS
FOR

DR32COM
FOR

OPCODES
FOR

VOLMOUNT
FOR

ERFSUMM
MAP

EMBLMDEF
FOR

QIOCOMMON
FOR

ERFPROC5
MAP

DEVERRDEF
REQ

RECSELDEF
REQ

BRIEF
LIS

EMBSPDEF
FOR

0147 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

CLASSIFY
LIS

DR750
LIS

CSTRING
LIS

DHEADS
LIS

DR780
LIS

DR11W
LIS

COMPRESS
LIS

DTAILS
LIS

DECODECC
LIS

DUMPREG
LIS

DUTUDRIVR
LIS

CALCMAP
LIS

DUP3271
LIS

CRYPTK
LIS

DQDISKS
LIS

DUP11
LIS